

Version

1

Batavia XBRL ▶▶



www.batavia-xbri.com

XBRL in Plain English

A SIMPLIFIED VIEW ON XBRL

WWW.BATAVIA-XBRL.COM

XBRL™ is a trademark of the American Institute of Certified Public Accountants ('AICPA')

© 2006, 2007 Batavia XBRL BV all rights reserved
Postal box 258, 2800 AG, Gouda
Phone +31 182 686 816 • Telefax +31 182 686 206

The contents of this publication are protected by Dutch copyright law and international treaties. Unauthorized reproduction of this publication or any portion of it is strictly prohibited.

While every precaution has been taken in the preparation of this book, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this book or from the use of programs and source code that may accompany it. In no event shall the publisher and the authors be liable for any loss of profit or any other commercial damage caused or alleged to be caused directly or indirectly by this book.

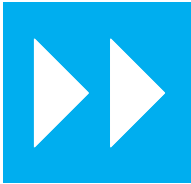
Version : 1
Revision : 1
Authors : Jos van der Heiden

Index

1	INTRODUCTION	1
1.1	WHAT TO EXPECT	1
1.2	INTRODUCING XBRL	2
1.2.1	Business Reporting	2
1.2.2	Extensible	4
1.2.3	Language	5
2	WHAT IS XBRL?	6
2.1	WHAT IS A TAXONOMY?	6
2.2	WHAT IS AN INSTANCE DOCUMENT?	7
3	TAXONOMY ANATOMY	9
3.1	TAXONOMY SCHEMA	10
3.1.1	Concept definitions	10
3.1.1.1	<i>Item data types</i>	10
3.1.2	Linkbase references	11
3.1.3	Roles and Arcroles	11
3.1.4	Example	12
3.2	LINKBASE	12
3.2.1	Common characteristics	14
3.2.1.1	<i>Arcs</i>	14
3.2.1.2	<i>Prohibiting and Overriding</i>	14
3.2.1.3	<i>Cycles</i>	14
3.2.1.4	<i>Roles</i>	15
3.2.2	Type-specific characteristics	15
3.2.2.1	<i>Label Linkbase</i>	15
3.2.2.2	<i>Reference Linkbase</i>	16
3.2.2.3	<i>Presentation Linkbase</i>	16
3.2.2.4	<i>Calculation Linkbase</i>	17
3.2.2.5	<i>Definition Linkbase</i>	17
4	AN INSTANCE IN TIME	18
4.1	THE XBRL ROOT ELEMENT	18
4.2	REFERENCES	19
4.2.1	Reference to taxonomy	19
4.2.2	Linkbase References	19
4.2.3	Role and Arcrole References	19
4.3	CONTEXT	19

4.3.1	Context Entity	19
4.3.2	Context Period	19
4.3.3	Context Scenario	19
4.4	UNIT OF MEASUREMENT	20
4.5	ITEM FACTS	20
4.5.1	Context Reference	20
4.5.2	Unit Reference	20
4.5.3	Precision or Decimals	20
4.6	TUPLE FACTS	20
4.7	FOOTNOTES	20
4.8	EQUALITY	21
5	EXPLORING NEW DIMENSIONS	22
5.1	INTRODUCTION	22
5.1.1	Dimensional Notions	23
5.2	DIMENSIONAL TAXONOMIES	24
5.2.1	Dimensions	24
5.2.1.1	<i>Typed Dimension</i>	24
5.2.1.2	<i>Explicit Dimension</i>	24
5.2.2	Domain-member relationships and inheritance	25
5.2.3	Hypercubes	25
5.2.4	Relating Primary Items to Hypercubes	26
5.2.5	Dimensional Relationship Sets	26
5.3	DIMENSIONS IN INSTANCE DOCUMENTS	27
5.3.1.1	<i>TypedMember</i>	27
5.3.1.2	<i>ExplicitMember</i>	27
5.3.2	Validation	28
5.3.2.1	<i>Validation of primary item facts</i>	28
5.3.3	Dimensional Equality	28
6	A DIVE INTO THE XBRL WATERS	29
6.1	DAY 1 – STARTING OUT	29
6.1.1	Taxonomy Schema Document	29
6.1.1.1	<i>Schema Root</i>	29
6.1.1.2	<i>Importing the XBRL Instance Schema</i>	30
6.1.1.3	<i>Concepts</i>	30
6.1.1.4	<i>Linkbase References</i>	32
6.1.2	Label Linkbase	32
6.1.2.1	<i>Locators</i>	33
6.1.2.2	<i>Labels</i>	33
6.1.2.3	<i>Label Arcs</i>	34
6.1.3	Presentation Linkbase	34
6.1.3.1	<i>Locators</i>	34
6.1.3.2	<i>Presentation Arcs</i>	35
6.1.4	Instance Document	36
6.1.4.1	<i>Taxonomy Schema Reference</i>	36
6.1.4.2	<i>Contexts</i>	36
6.1.4.3	<i>Units</i>	37
6.1.4.4	<i>Facts</i>	37
6.1.5	We proudly present to you: our instance document	38
6.2	DAY 2 – REFINING OUR INITIAL EFFORTS	39
6.2.1	Setting up sections	39
6.2.1.1	<i>Custom Roles</i>	39

6.2.1.2	Abstract Root Concepts	40
6.2.1.3	Separate PresentationLinks	40
6.2.1.4	Let's have Another Look	42
6.2.2	Back to the drawing board	43
6.2.2.1	Custom Roles	43
6.2.2.2	Abstract Concepts	43
6.2.2.3	Labels	43
6.2.2.4	Multilevel PresentationLink	44
6.2.2.5	Look at that!	45
6.2.3	Sections revisited	45
6.2.3.1	Custom Roles	45
6.2.3.2	Abstract Concepts	45
6.2.3.3	Presentation Hierarchy	45
6.2.3.4	One Last Look Today	47
6.3	DAY 3 – COUNT ON IT!	47
6.3.1	Taxonomy Schema	47
6.3.1.1	Linkbase Reference	47
6.3.1.2	Custom Roles	48
6.3.2	Calculation Linkbase	48
6.3.2.1	Custom Role References	48
6.3.2.2	Calculation Link	49
6.3.2.3	Locators	49
6.3.2.4	Calculation Arcs	49
6.3.3	What's the grand total?	50
6.3.3.1	Correction	51
6.4	DAY 4 – HOW TO MAKE LIFE EASIER	51
6.4.1	Structural Changes	52
6.4.2	So What Does It Look Like?	54
6.5	DAY 5 – OPENING UP NEW DIMENSIONS	55
6.5.1	The Primary Concepts	55
6.5.2	A Template Taxonomy	55
6.5.3	A Typical Dimension	56
6.5.4	Watch out: Explicit Language!	56
6.5.5	What's the Hype?	58
6.5.6	Presenting in dimensions	59
6.5.7	Dimensional instance	59
6.5.8	Our first peek in another dimension	63
6.6	DAY 6 – ENTERING MULTIPLE DIMENSIONS	64
6.6.1	Definition Linkbase	64
6.6.2	Instance document	65
6.6.3	When in doubt, use a Default	66
6.6.4	Never Ask A Lady About Her Age	66
6.7	CONCLUSION	68



1 Introduction

This chapter introduces this book and the main concepts from XBRL.

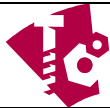
1.1 What to expect

If you start reading this book, you probably have already heard of a 'new' way to do business reporting called XBRL™. If you've taken a look at the specification of the XBRL language you'll know that it consists of a 158 page document full of formal definitions.

Such a document is necessary to correctly define XBRL. Perhaps it is even relaxing bed-time reading material to mathematicians. But for us 'normal' people it is less so.

For us 'normal' people, this book gives a 'plain English summary' of the XBRL specification. It should give you a good feel for what XBRL is and how it can be used. The book mainly focuses on the 'functionality' of XBRL as expressed by the specification.

You will **not** learn every nitty-gritty detail by reading this book. If you need that level of detail, e.g. when you want to write your own XBRL validating software, you must still refer to the formal specification. But even then, this book will certainly serve as an introduction to the exciting world of XBRL.



You can recognize the cases where I did feel I needed to go into more detail by this 'nuts & bolts' icon.

I will also **not** discuss most of the underlying technical standards, such as XML, XML Schema, XLink, XPath, XPointer etc. If you don't feel comfortable with these technologies, please refer to the World Wide Web Consortium (W3C) website at www.w3c.org or any good book on XML.

This book is based on XBRL version 2.1 recommendation 2003-12-31 plus the corrected errata of 2005-04-25. Whenever this book and the official specification disagree, modesty requires me to assume I made a mistake and the authors of the specification got it right. I'd recommend you to make the same assumption.



Examples used in this book are indicated by this image.

1.2 Introducing XBRL

XBRL stands for 'Extensible **B**usiness **R**eporting **L**anguage' which pretty much defines what it is: a language for reporting used in business. And that language is extensible.

It's easy, no? Well, perhaps it could bear a little bit more explanation.

(note: within this chapter some XBRL related terms are introduced. They can be recognized by the **bold** face. Subsequent chapters will explain each term, so don't worry about strange sounding words yet)

Let's jump right into the middle of it: ..."**B**usiness **R**eporting"...

1.2.1 Business Reporting

We all know there is a lot of reporting going on in business:

- tax returns;
- annual reports;
- inter-departmental sales figures;
- ...

Each report contains data, which is simply a number of 'facts' about that which is reported on, such as:

- reporting period;
- annual income;
- number of customers;
- number of sales;
- inventory numbers;
- ...

In the good old days, such reports were created by gathering all the relevant facts and having them typed on a pre-printed paper form. The typed up form was then sent to interested parties who could read the relevant facts from the form.

This sounds cumbersome, and it is. But it gets even worse...

Different parties tend to require data in different forms, but the underlying facts can easily be the same. This requires a reporter to gather those same facts into different aggregations for each form.

XBRL tries to provide a way to improve creating, sharing and using data in business reports. It defines an electronic format for reporting, enabling computers to create, validate and process reports automatically. It also defines a way to provide a common definition of the meaning of the reported 'business facts'. The reporter could simply make one report of the facts, leaving it up to the receiver of the report to select the relevant facts, combining them in any way that suits the receiver. The common definition ensures that each receiving party interprets the reported facts the same way.

Another interesting aspect is the distinction between the pre-printed fields on a form and the typed-in facts. The pre-printed form is a 'template' that *defines* what needs to be reported. This layout will be defined once by the party receiving the report. The typed-in facts are the *contents* of a report, created each time a report is made.

The XBRL standard also uses this distinction:

- A definition of what needs to be or can be reported is described in a so-called **taxonomy**. The taxonomy defines 'concepts' within the business realm on which can be reported.
- An actual report is called an **instance document**. This document contains the facts that are reported. A document refers to a taxonomy to give meaning to the facts. Within the document each fact is linked to its corresponding concept in the taxonomy.



This seems to be a good time to introduce an example I will be using throughout the book. It illustrates the concepts of XBRL and places the more technical / formal aspects in a practical context. The example consists of a pre-printed form and some handwritten reports.

The pre-printed form looks like this:

FA001		-	Company demographics	
<hr/>				
Company name	:	_____		
Reporting period	:	_____ - _____		
<hr/> <hr/>				
Totals				
1	Start of period			
1.a	Total number of employees	:	_____	
2	End of period			
2.a	Total number of employees	:	_____	
<hr/>				
Gender				
3.	Start of period			
3.a	Men	:	_____	
3.b	Women	:	_____	
4.	End of period			
4.a	Men	:	_____	
4.b	Women	:	_____	
<hr/>				
Age				
5.	Start of period			
5.a	.. - 20	:	_____	
5.b	21 - 40	:	_____	
5.c	41 - ..	:	_____	
6.	Start of period			
6.a	.. - 20	:	_____	
6.b	21 - 40	:	_____	
6.c	41 - ..	:	_____	

The form can be uniquely identified by its identifier: FA001. The concepts asked to report upon are the company name, reporting period and number of employees at the start and end of the period. The reporter is also asked to split the number of employees between men and women and several age groups.



One handwritten report might look something like this:

FA001 - Company demographics	
Company name	: <u>Sample Inc.</u>
Reporting period	: <u>01-01-2005</u> - <u>31-12-2005</u>
Totals	
1 Start of period	
1.a Total number of employees	: <u>35</u>
2 End of period	
2.a Total number of employees	: <u>41</u>
Gender	
3 Start of period	
3.a Men	: <u>23</u>
3.b Women	: <u>12</u>
4 End of period	
4.a Men	: <u>27</u>
4.b Women	: <u>15</u>
Age	
5 Start of period	
5.a .. - 20	: <u>5</u>
5.b 21 - 40	: <u>23</u>
5.c 41 - ..	: <u>7</u>
6 Start of period	
6.a .. - 20	: <u>9</u>
6.b 21 - 40	: <u>21</u>
6.c 41 - ..	: <u>11</u>

As can be seen, the number of employees has increased, but the company employs at least one person with lacking calculus skills. In this simple example it might be unlikely that anyone would miscalculate $27+15$ as 41, but in more complex forms such errors are highly likely when everything is done by hand.

1.2.2 Extensible

Another premise of XBRL is that it is extensible. Returning to the 'good old days', lets look at a scenario where extensibility would be useful:

Suppose the European Community defines a reporting requirement for any business within the EC.

- a. Such a requirement would most likely be stated in English, but most companies would like to have a form in their own language, since translating business concepts tends to be very tricky.
This would require separate versions of the same form in any language within the EC.
- b. Within some countries the government might already require part of the EC reporting, with some additions specific to the country.
To avoid having to send and receive two forms with overlapping requirements, both forms might be combined into one country-specific form. This would again require additional versions of the same form.

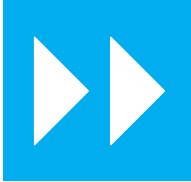
Again XBRL provides a way to support such requirements. The EC would create one taxonomy to define the reporting requirements. The translation from 'technical' concepts in the taxonomy to user readable 'labels' is contained in a so-called **presentation linkbase**. Each language within the EC could have its own linkbase or one linkbase might contain labels for each language. Note that the actual definition of concepts needs not be repeated in each language.

A country wanting to extend the EC taxonomy would simply create its own taxonomy which would refer to the EC taxonomy for all common concepts. The countries' taxonomy needs only to define the country-specific concepts.

1.2.3 Language

The 'L' in XBRL stands for Language. The XBRL language provides a way to express taxonomies and instance documents in one unambiguous format, which is a requirement to enable computers to handle documents.

The XBRL language is based on a number of 'world standards' such as XML and related specifications. The following chapters will explore this language in more detail.



2 What is XBRL?

Before diving into the XBRL specification itself, this chapter briefly introduces XBRL. As explained in the previous chapter, XBRL is actually about two things: defining what can be reported and actually reporting business facts. You will be introduced to both parts of XBRL: Taxonomies and Instances.

2.1 What is a taxonomy?

Remember the example of a pre-printed form I introduced in the first chapter? Here it is again:



FA001	-	Company demographics
<hr/>		
Company name	:	_____
Reporting period	:	_____ - _____
<hr/>		
Totals		
1	Start of period	
1.a	Total number of employees	: _____
2	End of period	
2.a	Total number of employees	: _____
<hr/>		
Gender		
3.	Start of period	
3.a	Men	: _____
3.b	Women	: _____
4.	End of period	
4.a	Men	: _____
4.b	Women	: _____
<hr/>		
Age		
5.	Start of period	
5.a	.. - 20	: _____
5.b	21 - 40	: _____
5.c	41 - ..	: _____
6.	Start of period	
6.a	.. - 20	: _____
6.b	21 - 40	: _____
6.c	41 - ..	: _____

The form exactly defines what needs to be reported as a set of 'entries' in which a company can write facts. In a form, the text before each entry defines what needs to be reported in the entry. You might also see footnotes to give more detailed information on an entry (e.g. if and how to count freelance contractors and part-time employees) or references to legislation, common practice, reference manuals and such.

Similar to this, a taxonomy is a document that defines a set of concepts on which can be reported. The taxonomy gives meaning and structure to the concepts and provides additional information in a number of ways:

- a. Concepts have a type, e.g. “cash” would be monetary (numeric with a denomination), while “name” would be plain text;
- b. Concepts can have relationships with each other (e.g. calculations) and with documentation (e.g. reference material);
- c. There are two types of concept:
 - Item
An item describes a single concept that can be reported on as an independently understood fact, e.g. “cash”, “number of employees”, ...
 - Tuple
A tuple is a ‘collection’ of other concepts (either item or tuple), grouping together related information that doesn’t have enough meaning by itself, e.g. a “manager” tuple could contain a “name” and a “title” item.

If you need companies to fill out a report, you could give such a ‘template’ to a printing company, order 10.000 copies, and every year send out forms to each company that must report to you.

Using XBRL you would simply direct each company to the taxonomy and ask to provide instance documents based on that taxonomy.

2.2 What is an instance document?

Let’s return to the example of a filled-in form as introduced in the first chapter:



FA001 - Company demographics	
Company name	: <u>Sample Inc.</u>
Reporting period	: <u>01-01-2005</u> - <u>31-12-2005</u>
Totals	
1 Start of period	
1.a Total number of employees	: <u>35</u>
2 End of period	
2.a Total number of employees	: <u>41</u>
Gender	
3 Start of period	
3.a Men	: <u>23</u>
3.b Women	: <u>12</u>
4 End of period	
4.a Men	: <u>27</u>
4.b Women	: <u>15</u>
Age	
5 Start of period	
5.a .. - 20	: <u>5</u>
5.b 21 - 40	: <u>23</u>
5.c 41 - ..	: <u>7</u>
6 Start of period	
6.a .. - 20	: <u>9</u>
6.b 21 - 40	: <u>21</u>
6.c 41 - ..	: <u>11</u>

As we saw in the previous section, a taxonomy basically defines the concepts on which can be reported, similar to the pre-printed form template.

By writing the required information onto the entries of a form, a report is made that can be filed. To read and process the report, you don't actually need the full template with all the explanatory text. Something like the following would be enough:

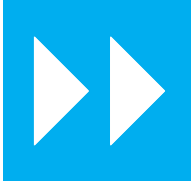


FA001
Sample Inc.
01-01-2005 - 31-12-2005
1a : 35
2a : 41
3a : 23
3b : 12
4a : 27
4b : 15
5a : 5
5b : 23
5c : 7
6a : 9
6b : 21
6c : 11

All you really need is some identification of the form and filer, the actual facts that are reported and for each fact an indication for which entry it is meant.

This is exactly what an XBRL instance document contains:

- A reference to the taxonom(y)(ies) on which the instance is based;
- Context identification;
- The facts, with references to the concepts in the taxonomy.



3 Taxonomy anatomy

This chapter with a title like an 60's rockband song describes the structure of an XBRL taxonomy. The focus is on **what** can be done with taxonomies and less on **how** it is done technically. We'll leave that level of detail for another chapter.

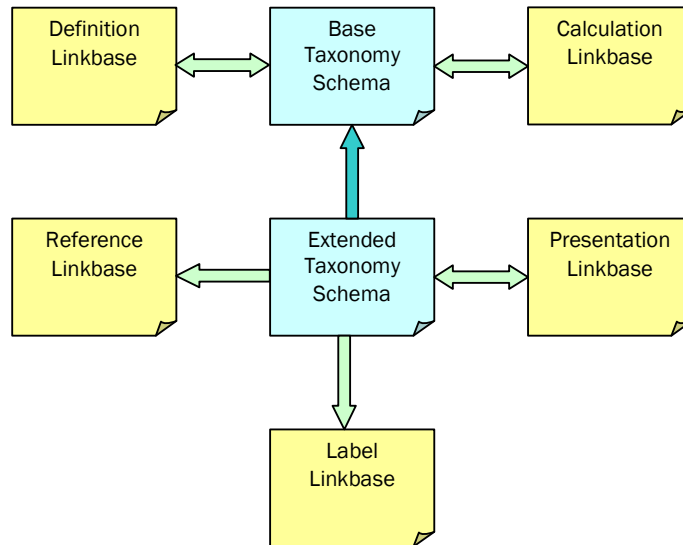
To start of: what we call **a** taxonomy is in reality usually a whole set of related documents called the **Discoverable Taxonomy Set** or **DTS** for short.


The starting point of a DTS is a taxonomy schema document. This is the taxonomy that an instance document references. This schema document can reference other documents, that can in turn reference other documents, and so forth. Discovering the taxonomy set means traversing each reference until all referenced documents are found


A taxonomy can reference two types of documents:

- Other taxonomies
This is the case when a taxonomy extends another taxonomy
- Linkbases
Linkbase documents are used to provide additional information about the concepts defined in the taxonomy. A Linkbase describes the relationships between concepts and other concepts and documentation. There are five different kinds of linkbase documents (each kind is discussed later on):
 - Definition
 - Calculation
 - Presentation
 - Label
 - Reference

Schematically an example of this can be visualized as follows:



 Note that some linkbases (Reference and Label) are 'uni-directional', i.e. there is only a link from the taxonomy to resources in the linkbase. Other linkbases (Definition, Calculation and Presentation) are 'bi-directional'. The links point from one part of the taxonomy to another part.

 If needed, a taxonomy must also refer to (or rather import) the XBRL instance schema, "xbrl-instance-2003-12-31.xsd", which defines all base elements needed to define concepts. This is one of those nitty-gritty details I'll be ignoring for most part in this book. I'll just assume that e.g. a taxonomy schema exists in a context, in this case the XBRL instance schema, and focus only on the more 'functional' aspects of XBRL.

The remainder of this chapter will describe each type of document in a little more detail.

3.1 Taxonomy Schema

The core of a taxonomy is its schema, which is a required document. This schema we're talking about is actually an 'XML Schema', a W3C standard 'language' in which the structure of XML documents can be defined. The XBRL specification takes XML Schema as the base language to define taxonomies. On top of this base language the XBRL specification defines set of XBRL specific additions and limitations.

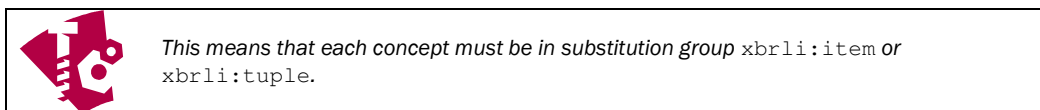
Within the schema it is possible to:

- import another 'base' taxonomy, enabling creating extensions of existing taxonomies;
- define concepts available for reporting;
- define custom roles used in linkbases;
- refer to linkbases.

3.1.1 Concept definitions

The XBRL specification states a number of requirements and recommendations on concept definitions:

- Each concept must have a unique name within the taxonomy schema;
- A concept must have a type attribute that specifies what kind of data can be reported on (see below);
- Concepts in a taxonomy schema are defined as either an item or a tuple;



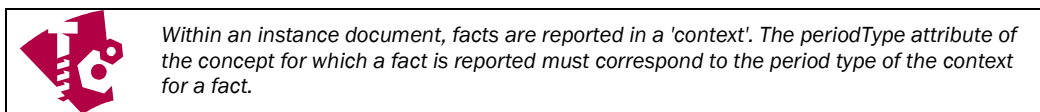
- It is recommended to give each concept a unique id attribute. This makes referring to the concept easier.

The XML Schema language allows for a number of ways to enhance the concept definition:

- It is possible to specify if a concept is mandatory or optional.
- Concepts can be defined to be 'abstract', which means you can not directly report on such a concept in an instance document. This is useful when extending concepts, as explained below, or when a root of a presentation hierarchy is needed.

In addition, XBRL defines two attributes for concepts: `periodType` and `balance`.

- The `periodType` attribute is required for items. It allows you to distinguish between concepts that are measurable at an instant in time and concepts that measure change over a period of time.



- If the type of a concept is monetary, the `balance` attribute may be used to specify if the concept defines a credit or debit value. This is useful for accounting concepts such as asset, liability, revenue and expense.

Concepts can be extended, creating a structured 'tree' of concepts. An example would be the general concept "manager" which could be extended as "business unit manager" and "division manager".

Since only non-abstract concepts may be reported on in an instance document, it is possible to prevent reporting of ambiguous facts.

In the example above, it might not be allowed to ambiguously report "manager" facts, but only the more specific "business unit manager" or "division manager". Making the "manager" concept abstract would accomplish this.

3.1.1.1 ITEM DATA TYPES

All item concepts must specify their data type. The types available are provided by the XML Schema language and the XBRL specification.

The types provided by XML Schema include boolean, text, decimal, date (-parts), etc. (see the XBRL specification section 5.1.1.3 for a full listing). Note: each type from XML Schema has a corresponding XBRL type definition.

The XBRL specification adds four additional types:

- monetary
This type must be used for concepts that represent monetary values.
- shares
Share-based concepts must use this type.
- pure
Concepts that represent measures where an implicit numerator and denominator is expressed in the same unit, such as growth rates, percentages, ...
- fractions
When the value of a fact cannot be represented exactly using any of the other types, e.g. 1/3 doesn't have an exact decimal representation, it can be reported as a fraction type. Fractions are given as separate numerator and denominator elements.

3.1.2 Linkbase references

The Taxonomy schema may, and usually does, contain references to Linkbases that provide additional information on the concepts in the taxonomy.

The reference may indicate the kind of links contained in the linkbase by defining a 'role' for the linkbase. The possible roles defined by the XBRL specification are:

- Definition
- Calculation
- Presentation
- Label
- Reference
- Footnote

Note: Footnotes are not contained in separate linkbases. The role is used only within instance documents.

It is also possible to define custom roles, which is another example of the extensibility of the XBRL specification.

To restrict the use of a linkbase even more, it is possible to define the type of facts that the links in the referred Linkbase may be used on.

3.1.3 Roles and Arcroles

Within linkbases, roles are used to identify different types of links. One generic role is defined by the XBRL specification to ensure all links have a role. Mostly though, the author of a taxonomy / linkbase will provide custom roles. Such roles are used to group relationships into so-called base sets. This segmentation of relations can be useful to e.g. identify different sections in a report when used in presentation linkbases. In calculation linkbases they are needed for any but the most simple calculations to ensure that calculations are not mixed up resulting in errors.

The relationships (as defined by arcs) themselves are typed by arcroles, that identify the type of relationship. A fair amount of generic arcroles, such as 'concept-label' and 'summation-item' are predefined by the XBRL specification. It is possible to define custom arcroles as well. But the purpose here is usually a technical one, e.g. to define new relationships in an extension on the XBRL specification. An example of this are the arcroles defined in the XBRL Dimensions specification discussed in chapter 5.

3.1.4 Example



Returning to the example 'taxonomy':

FA001 - Company demographics	
Company name	: _____
Reporting period	: _____ - _____
Totals	
1	Start of period
1.a	Total number of employees : _____
2	End of period
2.a	Total number of employees : _____
Gender	
3.	Start of period
3.a	Men : _____
3.b	Women : _____
4.	End of period
4.a	Men : _____
4.b	Women : _____
Age	
5.	Start of period
5.a	.. - 20 : _____
5.b	21 - 40 : _____
5.c	41 - .. : _____
6.	Start of period
6.a	.. - 20 : _____
6.b	21 - 40 : _____
6.c	41 - .. : _____

We could loosely 'define' the following concepts:

name	type
nr_employees_total	non-negative integer item
nr_employees_male	non-negative integer item
nr_employees_female	non-negative integer item
nr_employees_age_up_to_20	non-negative integer item
nr_employees_age_21_to_40	non-negative integer item
nr_employees_age_41_and_up	non-negative integer item

We'll elaborate on those concepts in the following section on Linkbases.

3.2 Linkbase

The concepts defined in a Taxonomy schema provide the 'bare building blocks' of the Taxonomy. They provide an exact listing what kinds of facts could be reported on.

Linkbases extend this definition in several ways:

- Elaborating on the definition with additional documentation, e.g. by giving a human readable label or a reference to literature.
- Describing the structure of concepts with inter-concept relationships, e.g. by defining calculation rules between concepts.



Let's first look at some possible linkbases for the example 'taxonomy', to get a feeling for what they are.

Note: the 'linkbases' given below are not complete or correct. They are meant illustrative only.

Label Linkbase

A label linkbase could assign an English label to each concept, as used on the form:

Concept	Label
nr_employees_total	Total
nr_employees_male	Men
nr_employees_female	Women

Another label linkbase could be used to assign Dutch labels to each concept:

Concept	Label
nr_employees_total	Totaal
nr_employees_male	Mannen
nr_employees_female	Vrouwen

Note: this example of a Label linkbase is not entirely correct. We will return to it later in the chapter.

Definition Linkbase

The definitions of the concepts provide different kinds of information. One definition might be that certain concepts require others:

Concept	Requires
nr_employees_male	nr_employees_female
nr_employees_female	nr_employees_male

If you specify either a male or female count, you must specify the other one as well.

If a 'nr_employees' concept were defined, the definition linkbase could specify that concept as the 'essence' of all other nr_employees_XXX concepts:

Essence	Alias
nr_employees	nr_employees_total
nr_employees	nr_employees_male
nr_employees	nr_employees_female

Calculation Linkbase

A calculation linkbase might define the following rule:

- $nr_employees_male + nr_employees_female = nr_employees_total$.

Such a rule would automatically catch the arithmetic mistake in our example filled in form.

Reference Linkbase

Suppose that the taxonomy describes a report which is required by law. The exact definition of what to report would be stated in that law, e.g. how to report on part-time employees.

Concept	Reference
nr_employees_total	Common Law; book IV; 156-32;b
nr_employees_male	Common Law; book IV; 156-32;b
nr_employees_female	Common Law; book IV; 156-32;b

Note: this would be expressed as a 'many-to-one' relationship, as discussed below.

Presentation Linkbase

The presentation linkbase describes how the concepts could be hierarchically presented in a report form. Our example could define the following hierarchy:

Concept	Children
abstract_group	nr_employees_total nr_employees_male nr_employees_female

3.2.1 Common characteristics

When you start looking at linkbases, you will see that they share a lot of common characteristics. The characteristics that are specific to each type of linkbase are described in the next section.

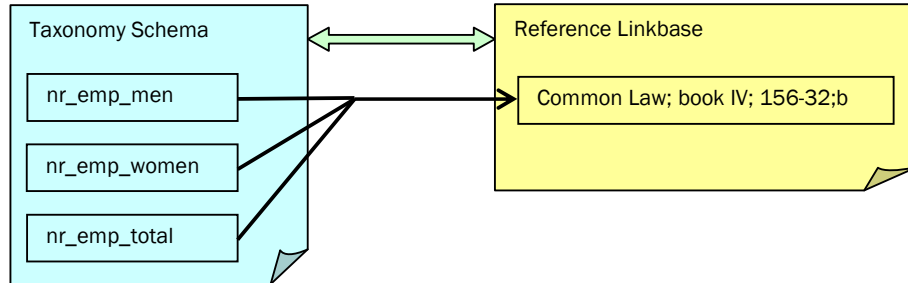
3.2.1.1 ARCS

All types of Linkbase use **'arcs'** to describe the relationships between concepts and resources or other concepts.

An arc is a two-sided, 'directional' relationship, which means the arc points 'from' one side 'to' the other side. Both the 'from' and the 'to' side of an arc may consist of one or more concept(s) and resources.



In our example of a reference linkbase, we could use a many-to-one arc like this to link all three concepts to the same reference:



The `to` and `from` attributes of an arc can be resources included in the extended link or locators in the extended link that point to concepts and resources in other documents.

A locator points to the concept or resource with a `href` attribute that may contain either an `id` of the pointed to concept or an XPointer expression that locates the concept or resource.

3.2.1.2 PROHIBITING AND OVERRIDING

When a taxonomy is extended, it may be needed to change the existing relationships between concepts. There are two mechanisms for this: **prohibiting** and **overriding**.

An arc may have a 'priority' attribute. When several arcs have a 'conflict', the one with the highest priority will prevail, in effect overriding the other arc(s).

When an existing arc is no longer valid, you may add a prohibiting arc in your own linkbase with a higher priority than the existing arc. This will prohibit the existing arc. A prohibiting arc is one where the optional `use` attribute has the value `'prohibit'`.



When applying the rules of overriding and prohibiting, the network of relationships that exists in the 'base' set is changed by the extensions.

Based on priority, new relationships can be added, existing can be removed or existing relationships can be replaced by new relationships.

The XBRL specification states that rules are applied to 'equivalent' relationships and it defines exactly when relationships are equivalent based on attributes of the arc and the 'from' and 'to' sides.

3.2.1.3 CYCLES

Arcs form a network of relationships. The network may become very complex. A cycle within the network occurs when there are two or more paths from one node to another node. The taxonomy must specify which types of cycles, if any are allowed:

- any
When any cycle is allowed, there are no limits on the relationships.
- undirected
This confusingly named option allows for cycles as long as there is no path back to a node. For example, nodes A

and B may have many paths where the direction is from A to B, but there may not be paths with a direction from B back to A.

- none
When no cycle is allowed, the network may have at most one path between any two nodes. This places a large restriction on the relationships.

Note: a path consists of a set of arcs between nodes. A path may be as short as one arc between two nodes, or as long as hundreds of arcs between hundreds of nodes.

3.2.1.4 ROLES

Each link in a linkbase can have one of several roles. The role defines the usage of the link. The available roles depend on the type of linkbase as described in the next section.

As said before, the XBRL specification provides a set of 'general purpose' roles for each type of linkbase. It is possible to define your own more specific roles to be used instead.

Roles can be used on resources and on arcs between them.

3.2.2 Type-specific characteristics

Although the different linkbases look very similar in structure, there are of course also differences. This section describes the specific characteristics for each type of linkbase.

3.2.2.1 LABEL LINKBASE

Labels provide human readable text in a specific language. Each resource must identify the language used in the label.

Labels can have several roles:

- label (this is the default role)
Used for standard labels
- terseLabel
used for short labels where parts of the description that can be inferred are omitted.
- verboseLabel
used for extended labels meant to give an exact description
- positiveLabel, positiveTerseLabel and positiveVerboseLabel
labels (standard, terse or verbose) used when a numeric fact has a positive value
- negativeLabel, negativeTerseLabel and negativeVerboseLabel
labels (standard, terse or verbose) used when a numeric fact has a negative value
- zeroLabel, zeroTerseLabel and zeroVerboseLabel
labels (standard, terse or verbose) used when a numeric fact has a value of zero
- totalLabel
used for labels on concepts that hold a total value
- periodStartLabel and periodEndLabel
used for labels on concepts presenting values associated with the start or end of a period.
- documentation
a label that provides documentation on a concept
- definitionGuidance, disclosureGuidance, presentationGuidance, measurementGuidance, commentaryGuidance and exampleGuidance
for labels that give an explanation on an aspect of the concept such as its definition, the way it should be measured or an example.

Label arcs have the role 'concept-label' which signifies it is always used to point **from** a concept **to** a label. The arcs can therefore never describe cyclic relationships between concepts.

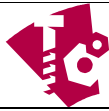


With this knowledge we could define a more complete label linkbase for some of the concepts in our example:

Concept	Language	Role	Label
nr_employees_total	en	totalLabel	Total
nr_employees_male	en	terseLabel	Men
nr_employees_female	en	terseLabel	Women
nr_employees_total	nl	totalLabel	Totaal
nr_employees_male	nl	terseLabel	Mannen
nr_employees_female	nl	terseLabel	Vrouwen

3.2.2.2 REFERENCE LINKBASE

References provide a way to link the definitions of concepts to authoritative statements published in business, financial and accounting literature. The element should provide only that information needed to identify and find the relevant publication. It should never contain the actual content of the publication.



References will be composed of parts, e.g. Common Law, book IV, article 342, part b.

The generic XBRL 'part' element is abstract, since the way publications are divided into parts varies for each publication / jurisdiction.

Again, the references can have several roles:

- reference (this is the default)
used for standard reference for a concept.
- definitionRef
used to reference a precise definition of a concept.
- disclosureRef, mandatoryDisclosureRef and recommendedDisclosureRef
used to reference an explanation of the disclosure requirements.
- unspecifiedDisclosureRef
used to reference an explanation of unspecified disclosure requirements, such as common practice, structural and completeness.
- presentationRef
reference to an explanation of the presentation of a concept.
- measurementRef
reference to the method for measuring values of the concept.
- commentaryRef
used to reference any other general commentary on the concept.
- exampleRef
reference to documentation that illustrates the usage of a concept by providing an example.

Reference arcs have the role 'concept-reference' which signifies it is always used to point **from** a concept **to** a reference. The arcs can therefore never describe cyclic relationships between concepts.

3.2.2.3 PRESENTATION LINKBASE

Presentation arcs have the role 'parent-child', which allows for the definition of a hierarchical structure of concepts. Both the 'parent' and 'child' side of the relationship must point to concepts.

When there is no concept that could naturally act as the root of a presentation hierarchy, it is possible to use an abstract concept defined specially for this purpose instead.

The arc can have a 'preferredLabel' attribute that identifies the preferred label **role** to use when presenting the child in the relationship. This is particularly useful when a concept is used several times in different ways within a DTS. Each presentation relationship could specify the appropriate label for that use of the concept.

The order attribute of the arc can be used to define the order in which concepts should be presented.

3.2.2.4 CALCULATION LINKBASE

Calculation arcs have the 'summation-item' role to represent aggregation relationships between concepts. The relationships point from one 'summation' concept to several contributing concepts. The arc relationship can only link item concepts.


Only items that are 'equal' with respect to context and unit to the summation will contribute to the summation.

Note: the relationships given by tuples also play a role here. It is for instance not possible to have items from duplicate tuples contributing to the same summation.

The arc has a 'weight' attribute that is used as multiplication factor for the item. This allows for more complex calculation relationships than simple addition. During multiplication, the necessary rounding should be performed.

The calculations within a DTS are all taken into account together. All calculationLinks with the same role on the same summation concept will be added together. If there are several breakdowns of the same value in one report, each calculation of the total should be placed in a calculationLink with its own specific role. Otherwise the different breakdowns would be added together leading to a multiple of the total being compared to the total fact.

An XBRL instance is consistent with the calculation linkbases if all calculations for the instance are consistent. This allows for automatic validation of a report with respect to the calculations.



The options provided by the calculation linkbase are very limited.

It is possible to make calculation that are more complex than simple summations, but that isn't easy and evenso the limitations are severe.

It is for example not possible to calculate the difference between facts for the same concept within different contexts, e.g. the end and start of a period.

Currently another linkbase, the Formula Linkbase, is being designed. This will become available in the near future and is meant to provide more extensive functionality for defining calculation formulas.

3.2.2.5 DEFINITION LINKBASE

The definition arc can have several roles, signifying different kinds of relationships:

- **general-special**
This relationship is between item concepts only and identifies generalizations / specializations of concepts. The relationship can be thought of as 'is-a-kind-of', for example: "division manager" is a specialization of "manager". Note that any valid value for a specialization item is also valid as value for the generalization item. The other way around usually doesn't hold.

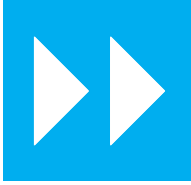
The network of general-special relationships may only contain 'undirected' cycles, as a specialization can never be its own generalization.

- **essence-alias**
This relationship can only be applied to item concepts. It is used when several 'similar' concepts are defined, identifying which is the 'best fit' or 'essence'. All other concepts are considered aliases of that essence concept.

The alias and essence concepts must have the same item type, the same period type and the same value for the balance attribute (if it is available).

The network of essence-alias relationships may only contain 'undirected' cycles, as an item can never be its own alias.

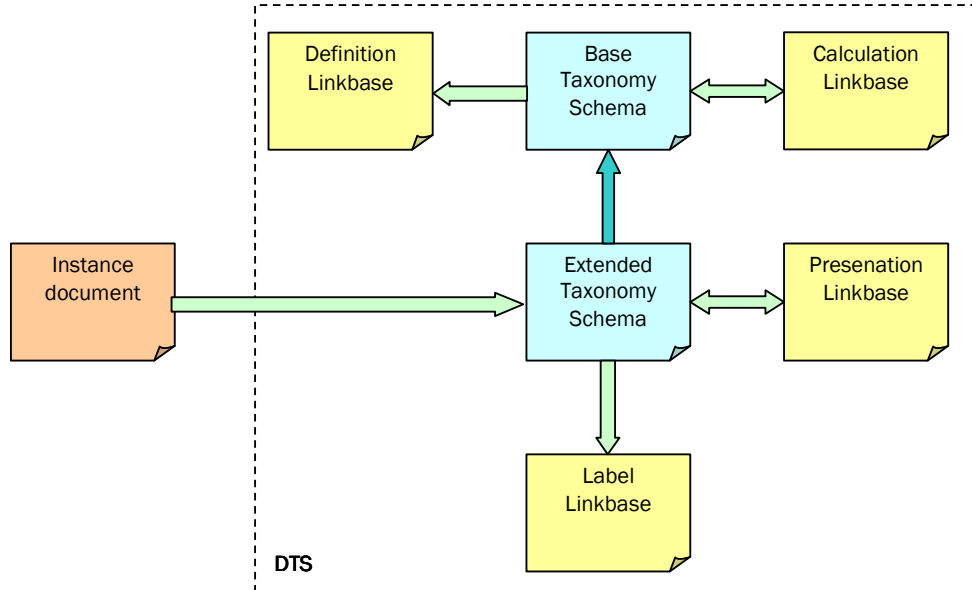
- **similar-tuples**
This relationship is between tuple concepts and is similar to the essence-alias role for item concepts. The relationship identifies tuples with equivalent definitions. An example would be a mailing address, which can take several formats, but holds equivalent data.
- **requires-element**
This relationship between concepts (tuples or items) identifies requirements within an instance document. If a fact of the 'from' concept occurs in the instance document, then a fact of the 'to' concept must also occur in the document.



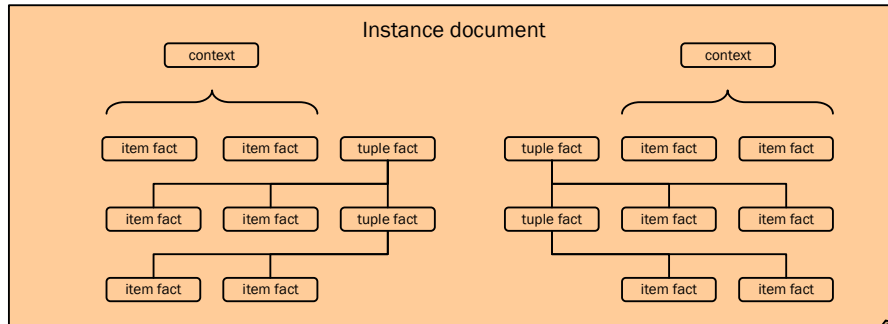
4 An instance in time

In this chapter we'll be looking at instance documents. Again, the focus is on the **what** instead of on the **how**.

An instance document contains the facts that comprise one report. The document refers to a taxonomy to lend meaning to the facts:



Facts can be either 'simple' item facts or compound tuple facts. All item facts within a document are reported in a context, e.g. the fiscal year or the start of a period. The document contains all available contexts for the report. Schematically the contents of an instance document could be shown as follows:



The following sections will describe the building blocks in some more detail.

4.1 The xbrl root element

An XBRL instance document contains different sorts of data:

- References to one or more taxonomies
- References to linkbases (optional)
- References to (arc)roles (optional)
- Context
- Units
- Facts (items and tuples)
- Footnotes

The single container for all of these is the *xbrl* root element that must be present in every instance document.

4.2 References

An instance document can contain several references to external resources.

4.2.1 Reference to taxonomy

The facts in the XBRL instance document are just random data without a taxonomy to give meaning to the facts. Each instance document must refer to at least one taxonomy. This reference should point to the taxonomy schema file.

4.2.2 Linkbase References

The author of an XBRL instance document can add references to linkbases that must become part of the DTS for the instance, together with the taxonom(y)(ies). This enables the author to provide additional information about the instance document.

Examples of this are additional definitions or references on concepts, custom presentation information or custom labels.

The way references are made from the instance document is identical to the way a taxonomy does this.

4.2.3 Role and Arcrole References

The XBRL instance may contain footnotes on the facts. If custom roles / arcroles are used for these footnotes, they should be referenced as well.

4.3 Context

To understand the facts reported in an instance document, they need to have a context. The context provided in XBRL contains the entity doing the reporting, the period on which is reported and optionally a 'scenario'.

A context must have an *id* attribute which is used to refer to the context from the items in the report.

4.3.1 Context Entity

The entity doing the reporting, i.e. the company, government department, individual, ..., is described by the context entity element.

The context entity contains an identifier that uniquely identifies the entity. This is done through providing a scheme and a 'token' that is a valid identifier in the namespace referenced by the scheme.

Examples are the NASDAQ ticker symbol or an D-U-N-S® number.

Optionally, the entity may also contain a 'segment' element to further specify which (part of an) entity is meant, e.g. a business unit within a company or a province within the state government.

XBRL does not state anything on what the segment element and content should look like, other than that it is XML and may not contain XBRL elements.

4.3.2 Context Period

The period on which is reported can in fact be either an instance in time or an actual period. When reporting on a period, it can have an explicit beginning and ending, or simply be 'forever'.

Note: when no time is specified, the time is implied to be at midnight. For the start date that is midnight at the start of the day. For end dates, midnight is taken to be at the end of the day, eventhough technically it is at the start of the next day.

Note: the end date must be after the start date.

4.3.3 Context Scenario

A report can have an optional scenario specified to indicate what type of facts are reported. Examples are actual, budgeted or restated figures.

The scenario describes the circumstances surrounding the measurement of the facts.

XBRL does not state anything on what the scenario element and content should look like, other than that it is XML and may not contain XBRL elements.

4.4 Unit of measurement

Numeric values must have a unit of measure to be able to interpret the value. Each unit used in an instance document must be provided as a unit element in the xbrl root element.

The unit contains either a single measure element, a product of measures or a ratio thereof.

Examples of units are EUR, meters, kilograms and square-feet.

A unit must have an *id* attribute which is used to refer to from items in the instance document.

4.5 Item Facts

Items are the single fact or business measurement on which is reported. An item may not contain other items. For structure, tuples must be used.

The content of the item may be either numeric or non-numeric. Apart from the actual content of the item, it also specifies additional data

4.5.1 Context Reference

Each item in the instance document must have a context which it refers to by *id*.

The period type of the item, instant or duration, must correspond to the type of period designated by the context:

- for instant items, the context must have an instance period
- for duration items, the context must have a valid startDate – endDate pair or the value ‘forever’

4.5.2 Unit Reference

Numeric items must state the unit of measurement used for its value. The unit is referred to by its *id*.

4.5.3 Precision or Decimals

Numeric items must indicate the accuracy of the value by providing either a ‘*precision*’ or a ‘*decimals*’ attribute. This accuracy is relevant when comparing values or performing calculations on values, especially when rounding or truncation must be done.

The precision attribute states how many digits in the value are significant. The decimals attribute states to how many decimals the given value is accurate. The two options are complementary ways to describe the accuracy. It is not allowed to include both attributes for the same item.



The precision of a value can be inferred when the decimals attribute is present.

The XBRL specification gives exact rules on how to infer the precision. It also provides examples on how to ‘read’ the attribute.

4.6 Tuple Facts

Most business facts can be understood independently of each other. These are provided as items in the xbrl root element. When facts can only be understood in relation with each other, a tuple is used to combine the facts into related sets.

An example would be the name and title of a manager.

Tuples group together concepts, both items and tuples are allowed as children of a tuple. The tuple itself may not refer to a context or unit. It also may not have a period type or balance attribute.

4.7 Footnotes

Items are understood independently and tuples may provide some structure to items that can only be understood in relation with each other.

Other, more irregularly structured, associations between facts and other relevant information are provided as footnotes.

Footnotes are links that are always included directly in the instance document. The role used for this is ‘footnote’. A footnote contains text and must have a language attribute.

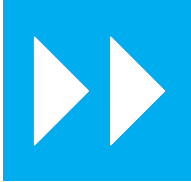
4.8 Equality

When comparing item or tuple facts, several forms of equality can be relevant:

- Identity
- Structure
- Parent
- Value
- XPATH
- Context
- Unit



The XBRL specification contains a large table that exactly specifies how each form of equality should be applied to different types of arguments, such as nodes, attributes, entities etc.



5 Exploring New Dimensions

The chapters in part one have shown you what XBRL is and what can be done with it. But as you already know, XBRL is extensible. The second part of this book looks at such extension modules. The first one is the XBRL Dimensions specification described in this chapter.

This chapter discusses the XBRL Dimensions version 1.0 specification, Candidate Recommendation dated 2006-06-19. At the time of writing this is still a candidate recommendation, but it is expected that the final version will not have any mayor surprises.

As with the XBRL specification itself, this chapter highlights the specification and introduces the most important ideas to give you a solid basic understanding of XBRL Dimensions. For the nitty-gritty details you should read the specification.

5.1 Introduction

Facts that are reported can usually be categorized. Examples are:

- Sales in various periods;
- Sales by product line;
- Sales by region;
- Sales by department;
- Number of employees by age;
- Number of employees by gender;
- ...

Two such categories are explicitly supported by the XBRL specification: period and entity (company, department, ...). These categories are always included in the contexts within which facts are reported.

It seems natural to want to extend this capability to enable taxonomy authors to specify their own categories, such as product line, gender, etc. This is exactly what the XBRL Dimensions specification does. Within this specification, categories are called dimensions.

The 'scenario' and 'segment' parts of a context as specified by XBRL can have any valid XML content. The Dimensions specification defines a formalized way to use these parts to apply new dimensions (categories) to contexts.



The example we have been using is very well suited to a dimensional approach: We only need to define a 'nr_employees' concept and two dimensions: 'gender' with values {'men', 'women'} and 'age group' with values {'... - 20', '21 - 40', '41 - ...'}.

The instance document contains a set of contexts for each period / dimension combination and each fact refers to one such context:

Context	Fact
01-01-2005	nr_employees = 35
01-01-2005 + [men]	nr_employees = 23
01-01-2005 + [women]	nr_employees = 12
01-01-2005 + [... - 20]	nr_employees = 5
01-01-2005 + [21 - 40]	nr_employees = 23
01-01-2005 + [41 - ...]	nr_employees = 7
31-12-2005	nr_employees = 41
31-12-2005 + [men]	nr_employees = 27
31-12-2005 + [women]	nr_employees = 15
31-12-2005 + [... - 20]	nr_employees = 9
31-12-2005 + [21 - 40]	nr_employees = 21
31-12-2005 + [41 - ...]	nr_employees = 11

5.1.1 Dimensional Notions

The XBRL Dimensions specification uses a number of notions that are briefly introduced here and explained in some more detail in the following section.

- **Dimension**
A Dimension is basically a categorization of facts. A Dimension is specified in a Domain Members Taxonomy and its members can be used in the contexts of an instance document to categorize the reported facts.
- **Domain & Domain Member**
The range of valid values for a Dimension is called its Domain. A Member of a Domain is one of these values. There are two types of members: typed and explicit.
Typed members are defined by syntactic constraints, such as 'integer numbers between 0 and 100'.
Explicit members are item concepts that are explicitly identified as member of a dimensions domain. This will be an 'enumeration' of members, such as 'men' and 'women' for the 'gender' dimension.
- **Hypercube**
Dimensions can be combined, e.g. sales by region by product line or number of employees by gender by age group. The set of possible values for such a combination of Dimensions can be visualized as points in a 'dimensional' space:
 - In one dimension the members form a section on a line;
 - With two dimensions the points form a square in a plane;
 - Three dimensions form a cube in space;
 - More dimensions cannot be easily visualized since we live in three-dimensional space.
 The 'mathematical' term for such forms is a 'hypercube'
- **Primary Taxonomy**
This is the 'normal' taxonomy as specified by XBRL. It defines the concepts on which can be reported. The XBRL Dimensions specification takes great care in ensuring that no changes on the primary taxonomy are required.
- **Domain Members Taxonomy**
The Dimensions and Dimension Members are defined as concepts in a 'Domain Members Taxonomy'.
- **Template Taxonomy**
The Template Taxonomy combines the Primary and Domain Members Taxonomies, specifying the structures of hypercubes and linking hypercubes to the primary item concepts they may be used on.

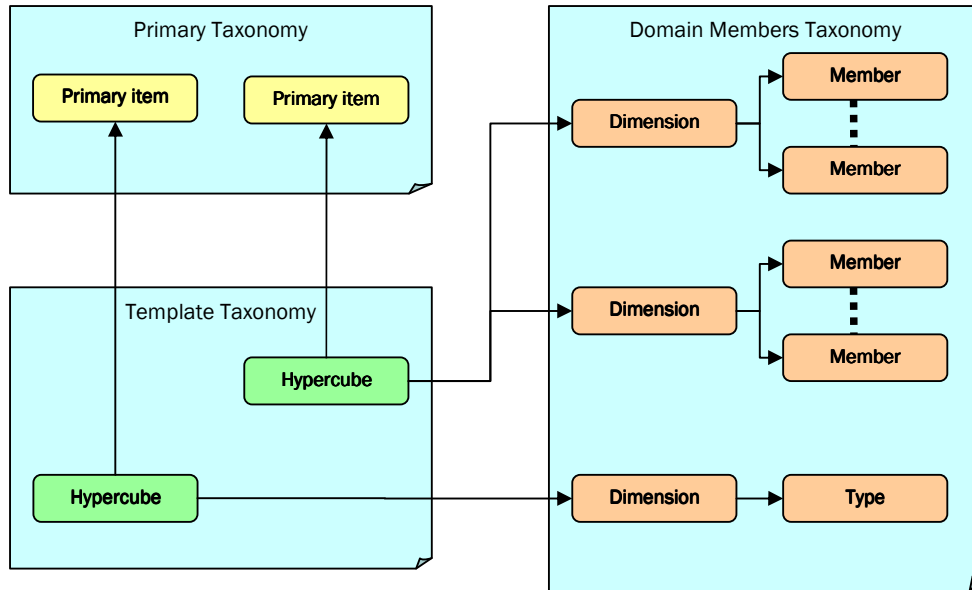


The specification defines three types of taxonomy. This distinction in three types is conceptual only.

It is possible to use three separate taxonomies, but the specification doesn't require separate taxonomies for each type. It is perfectly legal to combine types in one taxonomy. It is even possible for one concept to be used as primary item in one relationship and as e.g. a member item in another relationship.

5.2 Dimensional Taxonomies

This section explores the way taxonomies are extended to enable the specification of dimensions. I will start of with an



architectural diagram showing how the pieces fit together.

The following parts describe each dimensional piece of the architecture in some more detail.

5.2.1 Dimensions

A Domain Members Taxonomy defines Dimensions as abstract item concept declarations which must be in the *xbrldt:dimensionItem* substitution group.

Note: the *xbrli:balance*, *xbrli:periodType* and *nillable* attributes are ignored.

A dimension can be either 'typed' or 'explicit', which are two different ways to specify the domain of valid members.

5.2.1.1 TYPED DIMENSION

A typed dimension declaration must have a value for the attribute *xbrldt:typedDomainRef* that refers to an element declaration that defines the domain for the dimension.

The domain is specified using XML Schema types.

A *SimpleType* might e.g. specify integer values from 0 up to 100 to be valid or a String value consisting of 5 digits for a customer id

A *ComplexType* may also be used, e.g. an 'address' type with 'street', 'house number', 'zipcode' and 'city' elements.

5.2.1.2 EXPLICIT DIMENSION

The domain of an explicit dimension is identified by a *dimension-domain* relationship from the Dimension to the 'root' of a network of *domain-member* relationships between the Member elements.

An explicit dimension may not specify the *xbrldt:typedDomainRef* attribute.

The members of the dimension's domain are the qnames of all Member elements in the *domain-member* network.

A Member element is an item concept declaration in the *xbrli:item* substitution group (it may **not** be in the *xbrldt:hypercubeItem* or *xbrldt:dimensionItem* groups).

The *domain-member* relationships form a hierarchy of members. It is possible to include elements in the hierarchy, e.g. to provide structure as the root of a sub-hierarchy, that are not meant as members of the dimension's domain. The boolean

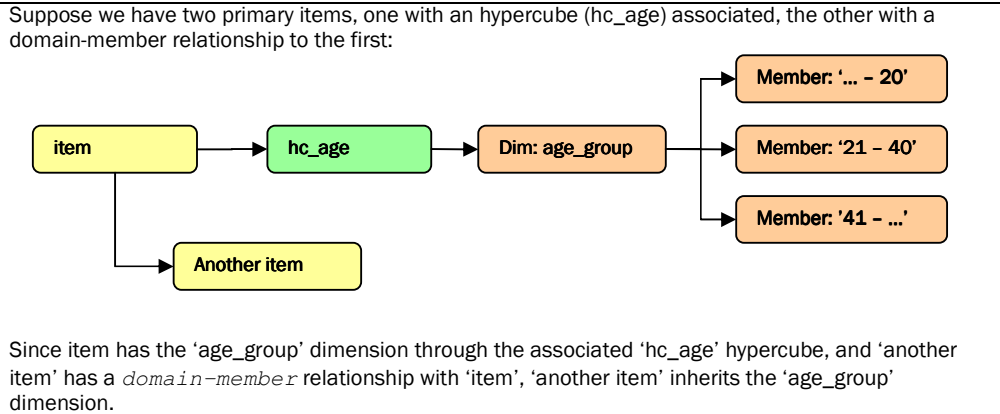
usable attribute, which has a default value of *true*, can be given the value *false* to indicate that the element pointed to is not a member of the domain.

An explicit dimension may be given have a default value through a *dimension-default* relationship to a domain member. This default value is inferred for a context when no value for a dimension is given. The default value may not be used as value within a context, it is always inferred!

Note that the *dimension-default* relationship does not automatically add the member to the domain of a dimension. It is not equivalent to the *domain-member* relationship.

5.2.2 Domain-member relationships and inheritance

Primary items may 'inherit' the hypercubes of other primary items by specifying a *domain-member* relationship between the primary items.



5.2.3 Hypercubes

Hypercubes are defined in a Template Taxonomy by combining zero (the hypercube may be empty) or more dimensions.

The declaring element is an abstract item concept declaration which must be in the *xbrldt:hypercubeItem* substitution group.

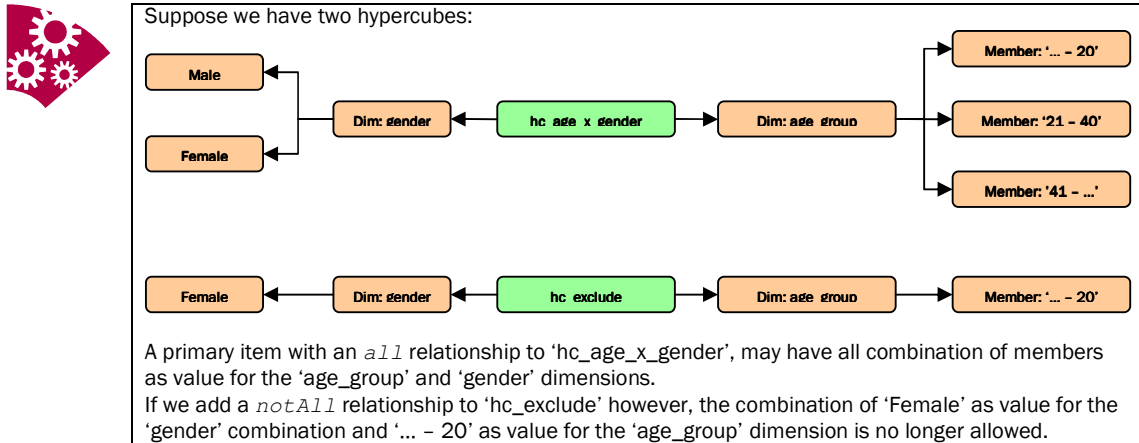
The Dimensions are related to the hypercube by arcs with the role *hypercube-dimension*. These relationships are ordered by the *order* attribute on each arc. The arcs may not define cyclic relationships.

5.2.4 Relating Primary Items to Hypercubes

As an author of a Dimensional Taxonomy you want to be able to control which concepts can have which dimensions. The Template Taxonomy provides for this by describing which hypercubes are allowed for which primary items by declaring a *has-hypercube* relationship between a primary item concept and a hypercube concept.

There are two types of *has-hypercube* relationships: *all* and *notAll*. The *all* relationship is used to define dimensions (and members) that are allowed for item facts. The *notAll* relationship defines dimensions and members that are not allowed.

Combining these relationships allows for fine-grained control of the dimensions and members allowed for each fact.



The *has-hypercube* relationship must specify in which part of the context the dimensions are to be defined: *segment* or *scenario*. The *contextElement* attribute must have one of these values.

The *segment* is used for dimensions that specify a part of the organizational structure of the entity, e.g. department, region, etc. The *scenario* is used for dimensions unrelated to the entities organizational structure, such as 'gender', 'product', etc.

The optional boolean *closed* attribute on a *has-hypercube* relationship can be given a value *true* to indicate that only values within the hypercubes dimensions are allowed. This attribute is only applied to the segment of scenario according to the value of the *contextElement* attribute of the relationship.

The default value is *false*, so not specifying the attribute will leave the segment or scenario open.

5.2.5 Dimensional Relationship Sets

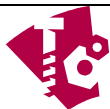
The relationships specified in the XBRL Dimensions are all included in definition linkbases. According to the XBRL specification, relationships are 'grouped' together in a network by taking all relationships from linkbases with the same role. This is called a base set.

The Dimensions specification extends the notion of a base set by introducing a *targetRole* attribute on several types of relationship (*all*, *notAll*, *hypercube-dimension*, *dimension-domain* and *domain-member*).

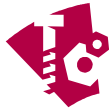
The *targetRole* refers to another role and indicates a transition from a linkbase in one base set to a linkbase in the base set indicated by the specified target role.

The set of relationships that are grouped together like this is called a Dimensional Relationship Set, or DRS.

Creating a DRS can be useful or even necessary when the dimensional taxonomy becomes more complicated. Without it you are very likely to include members in dimensions where they don't belong or add dimensions to hypercubes you didn't intend. You can easily end up with a set of relationships that make no sense or that may even contradict each other and be invalid.



This mechanism to go from one linkbase (role) to another makes validating a taxonomy or instance document a lot more complex, since relationships exist outside of the bounds of base sets as specified by XBRL.



The XBRL Dimensions specification uses the notion of 'consecutive relationships'. This means that e.g. members for a hypercube are found by following first the *hypercube-dimension* relationship and from each dimension found following the consecutive *domain-member* relationships.

Consecutive relationships are discovered within the same linkbase role if no *targetRole* attribute is specified.

Once a *targetRole* attribute is specified on an arc, the discovery of relationships moves on to the linkbase with the given role. There are no consecutive relationships discovered within the source linkbase (role).

The relationships that can be 'joined' as consecutive are limited to what you would logically expect: *has_hypercube* (all/ notAll) can have *hypercube_dimension* as consecutive relationship but not *dimension_domain* or *domain_member*, since that would 'skip' a relationship.

5.3 Dimensions in Instance Documents

As explained in the introduction, dimensions are specified in either the segment or scenario part of contexts in an instance document.

The choice between the segment and scenario part is made by the *contextElementType* attribute of the *has-hypercube* relationships.

5.3.1.1 TYPEDMEMBER

For a typed dimension, the value of a dimension is given as an *xbrldi:typedMember* child element within the segment or scenario.

The *dimension* attribute of this element must refer to the typed dimension declaration.

The contents of the *typedMember* is an element of the type of the dimension referred to by its *xbrldt:typedDomainRef* attribute. The value for the dimension is the value of that element.



Suppose we have a dimension 'ageDim', defined as an 'age' type with integer values ranging from 0 to (a bit optimistically) 150.

The dimension value of 45 is given as an 'age' child element within e.g. a segment as follows:

```
<xbrldi:typedMember dimension="d:ageDim">
  <d:age>45</d:age>
</xbrldi:typedMember>
```

5.3.1.2 EXPLICITMEMBER

An *xbrldi:explicitMember* element is used to specify a value for an explicit dimension. Again the *dimension* attribute of this element must refer to the explicit dimension declaration.

The value for the dimension is the contents of the explicitMember element and it must be the QName of one of the explicit members of the dimension.



Suppose we have a dimension 'ageGroupDim', with the following explicit members 'ageLessThan20', 'ageFrom21To40' and 'age41OrMore'.

The dimension value of age 21 - 40 is given as a child element within e.g. a segment as follows:

```
<xbrldi:explicitMember
  dimension="d:ageGroupDim">d:ageFrom21To40</xbrldi:explicitMember>
```


5.3.2 Validation

The XBRL Dimensions specification adds a set of validation rules to the XBRL specification.

5.3.2.1 VALIDATION OF PRIMARY ITEM FACTS

Primary item facts must be validated based on the item concept and the context for the item fact.

An item fact is automatically dimensionally valid if its concept has no *has-hypercube* relationships.

If the item concept does have *has-hypercube* relationships, the hypercubes that are referenced must be dimensionally valid. The context for the item fact must specify a valid combination of domain members or values for each referred dimension in the hypercube in at least one base set for the primary item.

If no value is given for a dimension, the default value of that dimension is inferred if it is available. It is not valid to specify more than one value for a dimension.

Note that attributes like *usable* on *domain-member* relationships and *closed* on *has-hypercube* relationships are taken into account when determining if the given value is valid within the dimension.



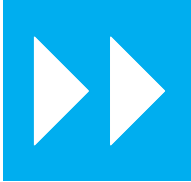
The XBRL Dimensions specification needs several pages to give a normative definition of dimensional validation, so the reality is a bit more complex than what I describe here.

However, the simple rules given above and using some common sense, should give you a pretty good understanding of what dimensionally valid means.

5.3.3 Dimensional Equality

The specification adds another type of equality to the already extensive list provided by the XBRL specification: 'd-equal'.

Two facts are considered d-equal for one dimension if they have the same value for that dimension.



6 A dive into the XBRL waters

In the previous chapters we have dipped our toes into the water, looking at what XBRL is and what can be done with it. With this knowledge we'll dive straight into the actual XBRL in this last chapter. Following the creation of a taxonomy and an instance document in several steps, we'll discover what XBRL looks like in 'real life'.

This chapter shows how a taxonomy and a corresponding instance document are created, based on the demographics report used as an example in the previous chapters.

Just like in real life, we won't be able to do everything right in our first try, so we'll have several different versions along the way. We'll be creating one version a day, which is only possible for a very small example. Most real-life taxonomies take several weeks or months (or even years) to create.

Note: we won't be making definition or reference linkbases. In real-life you would probably at least create one of these to define your concepts. For the purpose of this chapter they are not needed. Once you have mastered label linkbases and presentation linkbases, the definition and reference linkbases should be quite easy to understand.

6.1 Day 1 – Starting out

We start with a simple taxonomy for our example.

6.1.1 Taxonomy Schema Document

First of all we need a document in which the taxonomy schema is given. It is good practice to use the date the taxonomy is created in the file name to make sure different versions in time can be easily recognized.

Our project begins at the start of a new year, so we'll name our taxonomy document '*sample-2006-01-01.xsd*'. We will discuss the document in small parts, filling in the taxonomy one step at a time.

6.1.1.1 SCHEMA ROOT

Like any XML schema, we start with the root element '*schema*':

```
<schema
  targetNamespace="http://www.sample.com/2006-01-01"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xbrli="http://www.xbrl.org/2003/instance"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:sample="http://www.sample.com/2006-01-01">
</schema>
```


Notice the attributes within the root element:

- *targetNameSpace*
This defines the namespace of the elements in the schema. Schema aware software that is used to process the taxonomy or an instance referring to it should use this namespace to refer to the schema and its parts.

The other attributes define prefixes for elements in the schema document. The prefix identifies which namespace defines the element.

- *xmlns*
The default for all elements that don't have a prefix will be the xml schema namespace.
- *xmlns:xbrli*
The XBRL instance namespace elements get prefix *xbrli*.
- *xmlns:link*
The XBRL link namespace elements get prefix *link*.
- *xmlns:xlink*
Elements with prefix *xlink* reside in the xml schema xlink namespace.
- *xmlns:sample*
Lastly, any elements defined by ourselves, which are in the target namespace, will be prefixed with *sample*.

All documents used have similar attributes on the root element to define namespaces. We won't be describing these for each new type of document.




The XBRL specification includes a number of schemas:

- *Xbrl instance contains the definitions (types, attributes, elements etc.) needed to declare facts and concepts.*
- *Xbrl linkbase contains the definitions needed to declare links from one part of the taxonomy / linkbase to another*
- *Xbrl XLink (note the capitals) is used for declaring linkbases themselves. You would need this when creating your own extension to the XBRL specification.*
- *W3C xlink is the namespace for XML Link. Note that the schema for this specification is located at the www.xbrl.org website, since the w3c doesn't provide a schema and the XBRL specification requires a schema.*

6.1.1.2 IMPORTING THE XBRL INSTANCE SCHEMA

Our next step is to import the XBRL instance schema, so that we can use constructs from it in our own element definitions.

```
<import
  namespace="http://www.xbrl.org/2003/instance"
  schemaLocation="http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd" />
```



The import element must exactly locate the schema identified by the namespace. The schema document pointed to by the URI in the schemaLocation attribute must exist.

The XBRL instance schema imports the XBRL linkbase schema, and the xbrl linkbase schema in turn imports the XBRL XLink and W3C xlink schemas. This means we don't have to import these schemas ourselves.

The W3C XML Schema namespace is standardized and automatically recognized by a schema aware processor.

6.1.1.3 CONCEPTS

The concepts in our taxonomy are defined as elements in the schema. From the example form we earlier derived the following concepts:

Name	type
nr_employees_total	non-negative integer item
nr_employees_male	non-negative integer item
nr_employees_female	non-negative integer item
nr_employees_age_up_to_20	non-negative integer item
nr_employees_age_21_to_40	non-negative integer item
nr_employees_age_41_and_up	non-negative integer item

The total number of employees concept is defined as follows:

```
<element
  id="sample_nr_employees_total"
  name="nr_employees_total"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />
```

The attributes of the element are:

- *id*
This identifier is used to refer to the concept in other documents, such as linkbases. It must be unique within the schema. We have taken the name of the concept with a *'sample_'* prefix to try to make it globally unique.
- *name*
The name of the element identifies the concept within the taxonomy and instance documents. It must be unique within the taxonomy.
- *xbrli:periodType*
Each concept must have a period type associated with it. In our example, the number of employees is reported at the start and the end of a period, so at a specific date. The concepts have period type *instant* to signify this.
- *type*
The number of employees is taken to be an integer number, part-timers are counted as one and not as a fraction. Since there can never be a negative number of employees, we use the *nonNegativeIntegerItemType* as defined in the XBRL Instance schema.
- *substitutionGroup*
The concept is an item, so it must be in the *item* substitution group defined in the XBRL Instance schema.
- *nillable*
It is recommended to always define concepts as *nillable*.

The other concepts are defined exactly the same, off course with their own unique *id* and *name* attributes.

We will also need an abstract element to serve as root of the presentation hierarchy of our report. A abstract concept is given the (arbitrary) type *xbrli:stringItemType*. The concept can never be used in a fact, so we only need to supply the type because of a requirement in the XBRL specifications.

```
<element
  id="sample_presentation_root"
  name="presentation_root"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  abstract="true"
  nillable="true" />
```

An abstract concept must have an *abstract* attribute with the value *true*.

6.1.1.4 LINKBASE REFERENCES

If we want to present the taxonomy or instance documents, we need to provide additional information in the form of linkbases. We'll start with two linkbases: label and presentation:

The references to linkbases must be placed in an *appinfo* element, which must be placed in an *annotation* element.

```
<annotation>
  <appinfo>
    <link:linkbaseRef
      xlink:type="simple"
      xlink:href="sample-2006-01-01-label.xml"
      xlink:role="http://www.xbrl.org/2003/role/labelLinkbaseRef"
      xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase" />
    <link:linkbaseRef
      xlink:type="simple"
      xlink:href="sample-2006-01-01-presentation.xml"
      xlink:role="http://www.xbrl.org/2003/role/presentationLinkbaseRef"
      xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase" />
  </appinfo>
</annotation>
```

The linkbase references have the following attributes:

- *xlink:type*
The type of linkbase references is always 'simple'.
- *xlink:href*
The actual reference is given in this attribute. In our example, we refer to the presentation and label linkbases discussed in the next sections.
- *xlink:role*
The role attribute determines the type of linkbase that is referred to. We use the *presentationLinkbaseRef* and *labelLinkbaseRef* roles defined by the XBRL Linking schema.
- *xlink:arcrole*
The role of the arc used for linkbase references is always the *linkbase* role defined by XLink.

6.1.2 Label Linkbase

A Label Linkbase contains three types of elements:

- locators to identify the concepts within the taxonomy that are given a label
- label resources that contain the actual labels
- label arcs that link a concept (through its locator) with a label.

These elements are placed within a *labelLink* element.

```
<?xml version="1.0" encoding="UTF-8"?>
<linkbase
  xmlns="http://www.xbrl.org/2003/linkbase"
  xmlns:xbrli="http://www.xbrl.org/2003/instance"
  xsi:schemaLocation="http://www.xbrl.org/2003/instance
    http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:sample="http://www.sample.com/2006-01-01">
  <labelLink xlink:type="extended" xlink:role="http://www.xbrl.org/2003/role/link">
  </labelLink>
</linkbase>
```

A *labelLink* has two attributes:

- *xlink:type*
The type of linkbases is always 'extended'
- *xlink:role*
The default role of linkbases is that of *link*.

6.1.2.1 LOCATORS

Locators are placed in *loc* elements. They are used to locate the concepts in a taxonomy schema that are linked to.

```
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-01.xsd#sample_nr_employees_total"
  xlink:label="concept_nr_employees_total" />
```

- *xlink:type*
The type of a *loc* element is '*locator*', signifying it is used as a locator (of concepts).
- *xlink:href*
This attribute contains the actual location of the concept. Note that we use the *id* attribute of the concept to point to it.
- *xlink:label*
The *loc* element is given a *label* attribute to refer to within the linkbase. We used the convention to use the name of the concept with the '*concept_*' prefix as label.

Each concept that is given a label within the linkbase is pointed to with a *loc* element.

6.1.2.2 LABELS

Each label is included as the value of a label element.

```
<label
  xlink:type="resource"
  xlink:label="label_nr_employees_total"
  xlink:role="http://www.xbrl.org/2003/role/label"
  xml:lang="en">Total number employees</label>
<label
  xlink:type="resource"
  xlink:label="label_nr_employees_total"
  xlink:role="http://www.xbrl.org/2003/role/terseLabel"
  xml:lang="en">Total</label>
<label
  xlink:type="resource"
  xlink:label="label_nr_employees_total"
  xlink:role="http://www.xbrl.org/2003/role/verboseLabel"
  xml:lang="en">Total number of employees</label>
```

- *xlink:type*
The type of labels is '*resource*' indicating it is a local resource within the linkbase.
- *xlink:label*
Again, a *label* attribute is used to be able to refer to the label resource within the linkbase. Note that we give each label resource for the same concept the same value for the *label* attribute. This enables us to use a one-to-many link from a concept to all its labels. In this case three different types of label for the *nr_employees_total* concept.
- *xlink:role*
The role determines what type of label is given. The XBRL specification supplies a large number of predefined roles. In the code example above we have included three types: 'normal' label, terse label and verbose label. An application that renders a taxonomy or instance document can choose the most appropriate type of label. Presentation linkbases may also indicate which type of label is preferred.
- *xml:lang*
The language of the label must be given as attribute of the label element. The code example only includes English labels. It is possible to include labels for any needed language.

6.1.2.3 LABEL ARCS

Finally, after setting up locators to concepts and label resources, we can link concepts to labels using label arcs.

```
<labelArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/concept-label"
  xlink:from="concept_nr_employees_total"
  xlink:to="label_nr_employees_total" />
```

- *xlink:type*
The *type* of the *labelArc* element is 'arc' (who would have thought!)
- *xlink:arcrole*
The *arcrole* defines what kind of arc we are making. For labels it will be '*concept-label*': an arc between a concept and its label (or in our case its three different types of label).
- *xlink:from*
The *from* attribute points to one side of the arc. In this case the concept. We use the *label* of the locator to the concept as value for the attribute.
- *xlink:to*
Similarly, the *to* attribute points to the other side of the arc, the label resource, again using the *label* of the resource.
Since we used the same label for three different types of label, we create a one-to-many arc between the concept and its labels.

6.1.3 Presentation Linkbase

A Presentation Linkbase contains two types of elements:

- locators to identify the concepts within the taxonomy that are used in the presentation hierarchy
- presentation arcs that link a concept with another concept (through their locators).

These elements are placed within a *presentationLink* element.

```
<?xml version="1.0" encoding="utf-8"?>
<linkbase
  xmlns="http://www.xbrl.org/2003/linkbase"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xbrli="http://www.xbrl.org/2003/instance"
  xsi:schemaLocation="http://www.xbrl.org/2003/instance
    http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <presentationLink
    xlink:type="extended"
    xlink:role="http://www.xbrl.org/2003/role/link" >
  </presentationLink>
</linkbase>
```

6.1.3.1 LOCATORS

The locators are equivalent to those used in the label linkbase, so we won't describe them here again.

6.1.3.2 PRESENTATION ARCS

Presentation arcs link concepts into a parent-child hierarchy:

```
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_nr_employees_total"
  order="1" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/label" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_nr_employees_male"
  order="2" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_nr_employees_female"
  order="3" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
...
```

- *xlink:type*
Again, the value for type is 'arc'
- *xlink:arcrole*
For presentation links the arcrole is that of 'parent-child' which indicates that the presentation linkbase results in a hierarchical structure.
- *xlink:from*
The *from* attribute points to the parent concept in the relationship, through a *loc* element. Since we don't have a natural parent in our set of concepts, we have used the abstract *presentation_root* concept as parent for all other concepts.
- *xlink:to*
The *to* attribute points to the child concept in the relationship, through a *loc* element. Each concept in the taxonomy is put as a child under the abstract *presentation_root* concept.
- *order*
The *order* attribute determines the order of children when several concepts have the same parent.
- *priority*
Priority is used when overriding linkbases. We don't do that in our example (yet).
- *use*
By specifying 'optional' as value for the *use* attribute, we indicate that the concept may be part of the presentation network. It is common practice to specify this, though it is also the implied default when the attribute is not specified.
- *preferredLabel*
We have created several types of labels earlier in our label linkbase. Here we see how the presentation linkbase can indicate a preferred type of label for the child concept in the arc. This is a suggestion to software which type of label to use when rendering a taxonomy or instance document.
Note: since the preferredLabel can only be specified for the child concept, the root(s) of a presentation hierarchy can never be given a preferred label type.

6.1.4 Instance Document

Having created a taxonomy, we want to use it to create a report in an instance document.

An Instance Document contains several types of elements:

- reference to the taxonomy schema
- contexts for the facts
- units for numeric facts
- the actual facts themselves.

These elements are placed within an *xbrl* element.

```
<?xml version="1.0" encoding="utf-8"?>
<xbrl
  xmlns="http://www.xbrl.org/2003/instance"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:iso4217="http://www.xbrl.org/2003/iso4217"
  xmlns:s="http://www.sample.com/2006-01-01">
</xbrl>
```

6.1.4.1 TAXONOMY SCHEMA REFERENCE

The schema is referenced in a *schemaRef* element.

```
<link:schemaRef
  xlink:type="simple"
  xlink:href="http://www.sample.com/2006-01-01/sample-2006-01-01.xsd" />
```

- *xlink:type*
The type of reference is 'simple'.
- *xlink:href*
The reference is given as a *href* attribute. It points to the location of our taxonomy schema on the sample website.

6.1.4.2 CONTEXTS

We have two contexts for our sample report: the start of the reported period and the end of the period.

```
<context id="c_start">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
</context>
<context id="c_end">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
</context>
```

- *id*
The *id* is used to refer to contexts from within the facts that are reported. It must be unique within the instance document.
- *entity*
The entity that is reported on is placed in an *entity* element within the *context* element.
 - *identifier*
The *identifier* element uses a *scheme* to identify which entity is reported on. In our example we use a non-existing scheme within our own website.

- *period*
The *period* element contains the period or instance in time of the context as a child element.
 - *instant*
In our example we have two *instant* contexts for the start and end date of the period.



The scheme we used in the identifier element refers to a fictional statistics organization that publishes identifiers for companies. Our sample company has '12-34567' as identifier.

In the real 'real world' we would use e.g. a ticker symbol of a stock exchange or an identifier of a national tax authority or something similar.

6.1.4.3 UNITS

Numeric facts need a unit of measure. The units used in an Instance Document are presented as unit elements.

```
<unit id="u_person">
  <measure>Person</measure>
</unit>
```

- *id*
The *id* is used to refer to units from within the facts that are reported. It must be unique within the instance document.
- *measure*
The *measure* child element identifies the unit of measure. In our example we have defined our own measure: that of *Person*.

6.1.4.4 FACTS

Now finally we get at the goal of this whole exercise: reporting of facts.

```
<s:nr_employees_total
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">35</s:nr_employees_total>
<s:nr_employees_total
  contextRef="c_end"
  unitRef="u_person"
  decimals="0">41</s:nr_employees_total>

<!-- Gender Demographics -->
<s:nr_employees_male
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">23</s:nr_employees_male>
<s:nr_employees_male
  contextRef="c_end"
  unitRef="u_person"
  decimals="0">27</s:nr_employees_male>
<s:nr_employees_female
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">12</s:nr_employees_female>
<s:nr_employees_female
  contextRef="c_end"
  unitRef="u_person"
  decimals="0">15</s:nr_employees_female>

<!-- Age Group Demographics -->
<s:nr_employees_age_up_to_20
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">5</s:nr_employees_age_up_to_20>
<s:nr_employees_age_up_to_20
  contextRef="c_end"
  unitRef="u_person"
  decimals="0">9</s:nr_employees_age_up_to_20>
<s:nr_employees_age_21_to_40
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">23</s:nr_employees_age_21_to_40>
<s:nr_employees_age_21_to_40
```

```

contextRef="c_end"
unitRef="u_person"
decimals="0">21</s:nr_employees_age_21_to_40>
<s:nr_employees_age_41_and_up
contextRef="c_start"
unitRef="u_person"
decimals="0">7</s:nr_employees_age_41_and_up>
<s:nr_employees_age_41_and_up
contextRef="c_end"
unitRef="u_person"
decimals="0">11</s:nr_employees_age_41_and_up>

```

Each fact is an element with its name corresponding to the concept it belongs to. The value reported for the fact is the value of the element.

- contextRef*

Each fact refers to the context it belongs to, using the *id* attribute of that context. Our example contains facts for each concept within both contexts.
- unitRef*

Since we have all numeric facts, they must all reference a unit, in our case the *u_person* unit.
- decimals*

Numeric facts must state *decimals* or *precision*. Since we report on integer values only, we use *decimals* with a value of *0*.

6.1.5 We proudly present to you: our instance document

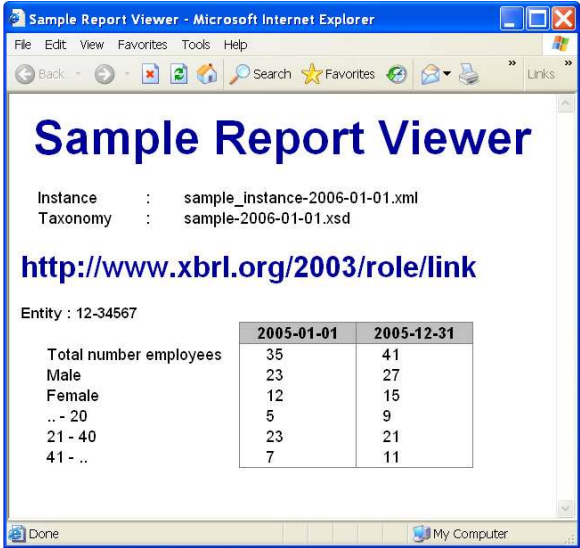
After all this work, we would like to see how the instance document could actually rendered, using the taxonomy, and the label and presentation linkbases in particular.

To do this we use a simple rendering tool that generates a web-page for the instance document, presenting facts in a table structure.

It uses the contexts in the document to create columns in the table. The labels, that are obtained from the label linkbase dynamically, are shown as leading first column.

The hierarchy specified in the presentation linkbase determines the ordering of facts in rows and indentation of the facts.

The result of our first days' efforts look something like this:



All in all not a bad result for one day's work!

6.2 Day 2 – Refining our initial efforts

OK, so we were very happy with the results of our initial efforts yesterday. But after a good night's sleep, we decide that it is not yet what we want.

The original form used different sections for totals, gender related demographics and age related demographics. And it didn't have that ugly 'http://www.xbrl.org/2003/role/link' title above the report either!

As it turns out, our renderer uses presentationLinks to define different sections and it prints the role from the presentationLink as the header of each section.

Yesterday we used the default role for our one presentation link, which explains the ugly title.

So the first thing to do is setting up sections with titles that mean something.

Note: we'll be using copies of yesterday's work where the date in the folder and filenames is changed from 2006-01-01 to 2006-01-02. Also, we'll be making several version today, so we'll make a subfolder for each version (/a, /b, /c etc.) and place the corresponding letter in all filenames. We only describe the changes with the previous day's version.

6.2.1 Setting up sections

Setting up sections involves several changes:

- Creating custom roles for the presentation linkbase, one for each section;
- Defining an abstract concept as root for each section;
- The presentation linkbase must include a presentationLink for each section.

6.2.1.1 CUSTOM ROLES

The rendering tool uses presentationLinks in our presentation linkbase to identify sections. The title for each section is derived from the role of the presentationLink. So we need to define custom roles in the taxonomy schema to hold the titles.

Our example form is divided into three sections: "totals", "gender" and "age", so we define a custom role for each section. These roles also are placed in the *appinfo* element within the *annotation* element.

```
<link:roleType
  roleURI="http://www.sample.com/totalEmployees" id="totalEmployees">
  <link:definition>Total number of employees</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
</link:roleType>
<link:roleType
  roleURI="http://www.sample.com/genderDemographics" id="genderDemographics">
  <link:definition>Gender related demographics on employees</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
</link:roleType>
<link:roleType
  roleURI="http://www.sample.com/ageDemographics" id="ageDemographics">
  <link:definition>Age related demographics on employees</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
</link:roleType>
```

The custom roles are defined as link:roleType elements with two attributes and two child elements:

- *roleURI*
The 'name' of the role is given in the form of an URI. We have used URI's within our own website, as is commonly done.
- *id*
The identifier attribute is used to refer to the custom roles from within other documents. It must be unique within the taxonomy schema.
- *link:definition*
This child element supplies a human readable definition of the role. When rendering a taxonomy or instance, it can be used as a title for each section, which is exactly what our simple rendering tool does.
- *link:usedOn*
This child element defines what type of linkbases may use the role. There may be several elements, but in our case only the presentation linkbase will use the custom roles.

6.2.1.2 ABSTRACT ROOT CONCEPTS

Instead of the one abstract concept we used yesterday, we define three abstract concepts for the three sections.

```
<element
  id="sample_section_totals"
  name="section_totals"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  abstract="true"
  nillable="true" />
<element
  id="sample_section_gender"
  name="section_gender"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  abstract="true"
  nillable="true" />
<element
  id="sample_section_age"
  name="section_age"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  abstract="true"
  nillable="true" />
```

Each abstract concept can be used as a root of one presentation hierarchy.

6.2.1.3 SEPARATE PRESENTATIONLINKS

The presentation linkbase needs to be updated as well. First of all, we need to make the custom roles available to the presentation links. We do this by making references to the roles.

```
<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-02a.xsd#totalEmployees"
  roleURI="http://www.sample.com/totalEmployees" />
<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-02a.xsd#genderDemographics"
  roleURI="http://www.sample.com/genderDemographics" />
<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-02a.xsd#ageDemographics"
  roleURI="http://www.sample.com/ageDemographics" />
```

- *xlink:type*
The type of reference is 'simple'.
- *xlink:href*
The *href* attribute points to the custom roles. Similar to the *href* attribute in *loc* elements, it uses the identifier of a role to point in the taxonomy schema document.
- *roleURI*
Within the presentation linkbase the roles are referred to by URIs. In our example we use the same URIs that were defined in the taxonomy schema itself.

Now we can make separate presentationLinks for each section.

```
<presentationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/totalEmployees" >
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02a.xsd#sample_section_totals"
    xlink:label="concept_section_totals" />
  </loc
/>>
```

```

        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_total"
        xlink:label="concept_nr_employees_total" />

<presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_section_totals"
    xlink:to="concept_nr_employees_total"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/label" />
</presentationLink>

<presentationLink
    xlink:type="extended"
    xlink:role="http://www.sample.com/genderDemographics" >

    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_section_gender"
        xlink:label="concept_section_gender" />
    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_male"
        xlink:label="concept_nr_employees_male" />
    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_female"
        xlink:label="concept_nr_employees_female" />

    <presentationArc
        xlink:type="arc"
        xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
        xlink:from="concept_section_gender"
        xlink:to="concept_nr_employees_male"
        order="1" priority="0" use="optional"
        preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
    <presentationArc
        xlink:type="arc"
        xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
        xlink:from="concept_section_gender"
        xlink:to="concept_nr_employees_female"
        order="2" priority="0" use="optional"
        preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>

<presentationLink
    xlink:type="extended"
    xlink:role="http://www.sample.com/ageDemographics" >

    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_section_age"
        xlink:label="concept_section_age" />
    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_age_up_to_20"
        xlink:label="concept_nr_employees_age_up_to_20" />
    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_age_21_to_40"
        xlink:label="concept_nr_employees_age_21_to_40" />
    <loc
        xlink:type="locator"
        xlink:href="sample-2006-01-02a.xsd#sample_nr_employees_age_41_and_up"
        xlink:label="concept_nr_employees_age_41_and_up" />

    <presentationArc
        xlink:type="arc"
        xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
        xlink:from="concept_section_age"
        xlink:to="concept_nr_employees_age_up_to_20"
        order="1" priority="0" use="optional"
        preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationArc>

```

```

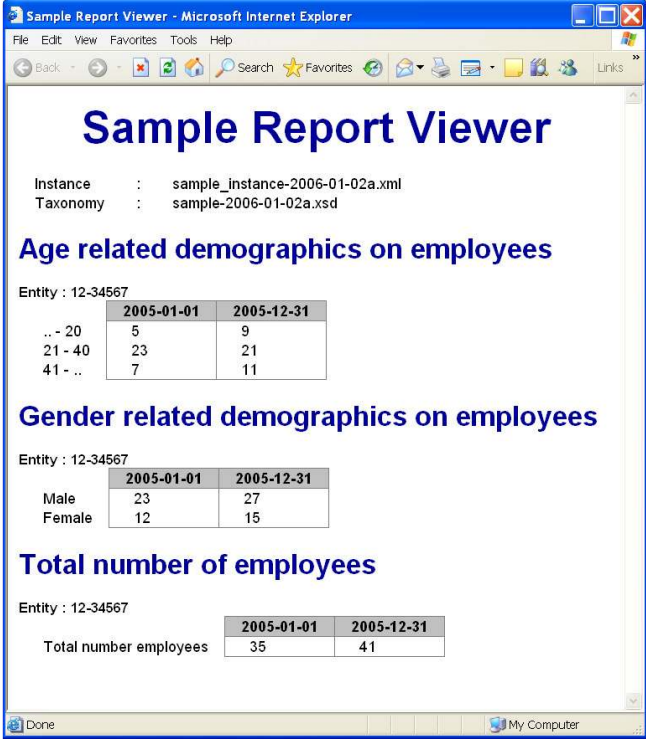
xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
xlink:from="concept_section_age"
xlink:to="concept_nr_employees_age_21_to_40"
order="2" priority="0" use="optional"
preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
xlink:from="concept_section_age"
xlink:to="concept_nr_employees_age_41_and_up"
order="3" priority="0" use="optional"
preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>

```

Each presentationLink uses the role corresponding to its own section. The relevant concepts for each section are located within the presentationLink, and a hierarchy is built with only the concepts belonging in each section.

6.2.1.4 LET'S HAVE ANOTHER LOOK

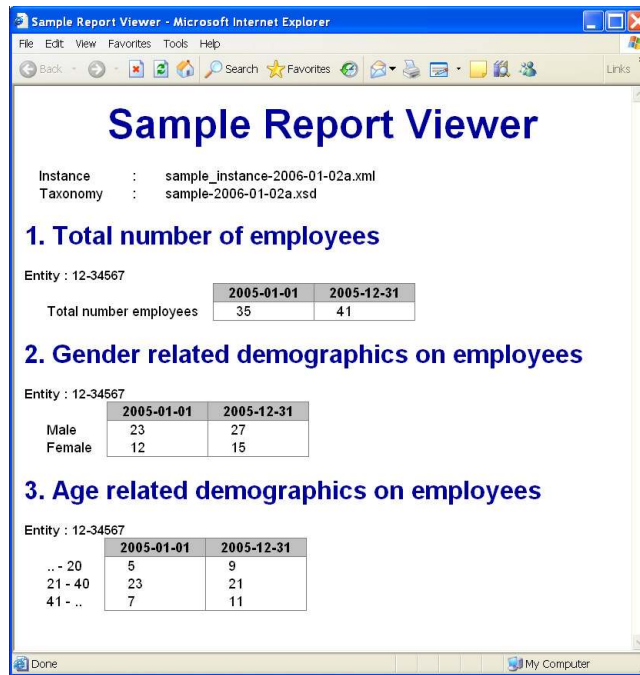
When we use our simple tool to render the instance document, we get a far better result. The report is divided into the three sections, each with its own title.



Note that we used another instance document. The content is the same as that of yesterday, the only difference being the taxonomy that was referred to.

One thing that is annoying however is the fact that the sections are not in the order we would like. Within a presentation hierarchy we can specify the ordering of children, but we cannot do the same for sections.

One option is to make sure the renderer gives the correct ordering. As it happens, our simple rendering tool sorts the sections alphabetically. By changing the descriptions of the custom roles we can force the order of sections.



This looks better, but it doesn't feel right to have to rely on some 'feature' of the rendering software. In real life this is not uncommon, but we will try and go back to the drawing board in the second try of this day to see if we can come up with a better solution.

6.2.2 Back to the drawing board

So, creating three sections did clear up some of the layout of the rendered report, but the overall result is still not what we wanted. Maybe we went a bit overboard.

There is another solution: the presentation link allows for a hierarchy of concepts, but up till now we have used only very 'flat' hierarchies: only one list of children beneath an abstract concept. Perhaps we should have a go at making a bit more of this hierarchy.

6.2.2.1 CUSTOM ROLES

The use of a custom role to give a meaningful title to the section was not such a bad idea, so for this version we'll keep one custom role.

```
<link:roleType
  roleURI="http://www.sample.com/employeeDemographics"
  id="employeesDemographics">
  <link:definition>Employees demographics</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
</link:roleType>
```

6.2.2.2 ABSTRACT CONCEPTS

We'll keep the abstract presentation_root concept, and we'll also use the abstract section concepts of the previous try. This should give us plenty of concepts to build a nice hierarchy with.

6.2.2.3 LABELS

Up till now, the abstract concepts didn't have labels. For the section concepts we'll add appropriate labels this time. This is done similar to all other labels, so I won't show examples here.

6.2.2.4 MULTILEVEL PRESENTATIONLINK

The presentation linkbase we use in this version specifies a hierarchy with more than one level.

```
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_section_totals"
  order="1" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/label" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_totals"
  xlink:to="concept_nr_employees_total"
  order="1" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/label" />

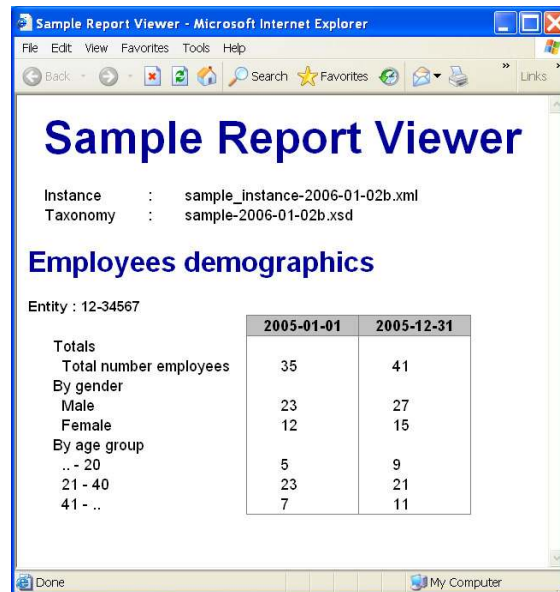
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_section_gender"
  order="2" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/label" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_gender"
  xlink:to="concept_nr_employees_male"
  order="1" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_gender"
  xlink:to="concept_nr_employees_female"
  order="2" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />

<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_presentation_root"
  xlink:to="concept_section_age"
  order="3" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/label" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_age"
  xlink:to="concept_nr_employees_age_up_to_20"
  order="1" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_age"
  xlink:to="concept_nr_employees_age_21_to_40"
  order="2" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
  xlink:from="concept_section_age"
  xlink:to="concept_nr_employees_age_41_and_up"
  order="3" priority="0" use="optional"
  preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
```

The abstract presentation_root concept has the three abstract section concepts as children. Each abstract section concept in turn has the appropriate concepts for that section as its children.

6.2.2.5 LOOK AT THAT!

The result of this second try looks like this:



	2005-01-01	2005-12-31
Totals		
Total number employees	35	41
By gender		
Male	23	27
Female	12	15
By age group		
.. - 20	5	9
21 - 40	23	21
41 - ..	7	11

This hierarchy does allow us to specify the exact layout and ordering of the concepts. But it still doesn't feel right. We had to go back to using just one section. What if somebody wanted to use only part of the report? I feel a third try coming on!

6.2.3 Sections revisited

In our third and last try of this day, let's think about sections for a bit. We started out today identifying three sections on the report form. But was that really correct?

Keep in mind that sections in XBRL reports tend to group together related information. Somebody could decide to use only one section and would still expect to have a meaningful report.

Perhaps in our simple example this doesn't really apply. But if you look at real-life taxonomies, you will find they tend to use sections for different views on the same data: an example would be a taxonomy with sections for balance sheet and profit/loss.

Would anyone interested in demographics want only the total number of employees? Perhaps, but not very likely. On the other hand: when looking at gender related demographics, you might also want to have the total numbers (imagine a more complex report with more than two concepts in the section).

So what if we split into two sections: gender demographics and age group demographics, each including the total numbers? That's what we'll do in the last try of today.

6.2.3.1 CUSTOM ROLES

We set up sections similar to our first try, but only for gender and age group.

6.2.3.2 ABSTRACT CONCEPTS

As you'll see in the next section, we won't need any abstract concepts anymore.

6.2.3.3 PRESENTATION HIERARCHY

Adding the totals to each section gives us natural candidates for the root of our presentation hierarchy.

```

<presentationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/genderDemographics" >
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_total"
    xlink:label="concept_nr_employees_total" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_male"
    xlink:label="concept_nr_employees_male" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_female"
    xlink:label="concept_nr_employees_female" />

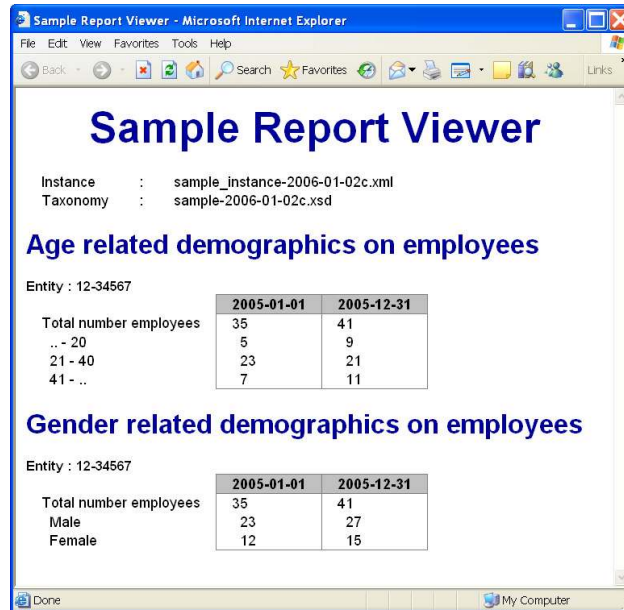
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_nr_employees_male"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_nr_employees_female"
    order="2" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>
<presentationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/ageDemographics" >
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_total"
    xlink:label="concept_nr_employees_total" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_age_up_to_20"
    xlink:label="concept_nr_employees_age_up_to_20" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_age_21_to_40"
    xlink:label="concept_nr_employees_age_21_to_40" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-02c.xsd#sample_nr_employees_age_41_and_up"
    xlink:label="concept_nr_employees_age_41_and_up" />

  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_nr_employees_age_up_to_20"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_nr_employees_age_21_to_40"
    order="2" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_nr_employees_age_41_and_up"
    order="3" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>

```

6.2.3.4 ONE LAST LOOK TODAY

Again, we look at the result of our effort:



I don't know about you, but I think this is a pretty good result. Each section is clear and self-contained. We'll leave it at this for today.

6.3 Day 3 – Count on it!

Now that we have a taxonomy that results in nicely rendered reports, it is time to start looking at another issue: validation.

As you may recall, our company employs one mathematically challenged book-keeper. Even though the report looks nice, 27 male employees and 15 female employees simply don't add up to 41 total employees.

Luckily XBRL has just the thing for us: calculation linkbases!

6.3.1 Taxonomy Schema

We need to make a few changes to the taxonomy schema to start using a calculation linkbase:

- Refer to the linkbase
- Allow the use of custom roles

6.3.1.1 LINKBASE REFERENCE

First of all we need to refer to the calculation linkbase to include it in the DTS.

```
<link:linkbaseRef
  xlink:type="simple"
  xlink:href="sample-2006-01-03-calculation.xml"
  xlink:role="http://www.xbrl.org/2003/role/calculationLinkbaseRef"
  xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase" />
```

We point to the document containing the calculation linkbase in the *href* attribute. The *role* attribute is set to *'calculationLinkbaseRef'* to indicate we are referencing a calculation linkbase.

6.3.1.2 CUSTOM ROLES

We have two calculations that can be made:

- the number of employees per gender should add up to the total number of employees
- and
- the number of employees per age group should add up to the total number of employees

Since we have only one total number of employees concept, we must use different roles for each calculation. Forgetting this would lead to calculations of twice the total number of employees (once per gender and once per age group).

Since we already have custom roles for our presentation linkbase, let's allow these to be used for calculations as well.

```
<link:roleType
  roleURI="http://www.sample.com/genderDemographics" id="genderDemographics">
  <link:definition>Gender related demographics on employees</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
  <link:usedOn>link:calculationLink</link:usedOn>
</link:roleType>
<link:roleType>
  roleURI="http://www.sample.com/ageDemographics" id="ageDemographics">
  <link:definition>Age related demographics on employees</link:definition>
  <link:usedOn>link:presentationLink</link:usedOn>
  <link:usedOn>link:calculationLink</link:usedOn>
</link:roleType>
```

Allowing the use of the role in *calculationLinks* simply means adding a *usedOn* child element with value 'calculationLink'.

6.3.2 Calculation Linkbase

The start of our linkbase document should start to look familiar by now.

```
<?xml version="1.0" encoding="utf-8"?>
<linkbase
  xmlns="http://www.xbrl.org/2003/linkbase"
  xmlns:link="http://www.xbrl.org/2003/linkbase"
  xmlns:xbrli="http://www.xbrl.org/2003/instance"
  xsi:schemaLocation="http://www.xbrl.org/2003/instance
    http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
</linkbase>
```

6.3.2.1 CUSTOM ROLE REFERENCES

Just like the presentation linkbase, we need to refer to the custom roles in the taxonomy schema.

```
<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-03.xsd#genderDemographics"
  roleURI="http://www.sample.com/genderDemographics" />
<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-03.xsd#ageDemographics"
  roleURI="http://www.sample.com/ageDemographics" />
```

Again, no surprises here.

6.3.2.2 CALCULATION LINK

We make a calculationLink element for each calculation, using the custom roles to differentiate between them.

```
<calculationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/genderDemographics" >
</calculationLink>
<calculationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/ageDemographics" >
</calculationLink>
```

6.3.2.3 LOCATORS

The locators are equivalent to those used in the label and presentation linkbases, so we won't describe them here again.

6.3.2.4 CALCULATION ARCS

And finally we can make calculation arcs, combining the concepts in each section to the total number of employees.

```
<calculationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
  xlink:from="concept_nr_employees_total"
  xlink:to="concept_nr_employees_male"
  weight="1.0"
  order="1" />
<calculationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
  xlink:from="concept_nr_employees_total"
  xlink:to="concept_nr_employees_female"
  weight="1.0"
  order="2" />
```

```
<calculationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
  xlink:from="concept_nr_employees_total"
  xlink:to="concept_nr_employees_age_up_to_20"
  weight="1.0"
  order="1" />
<calculationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
  xlink:from="concept_nr_employees_total"
  xlink:to="concept_nr_employees_age_21_to_40"
  weight="1.0"
  order="2" />
<calculationArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/2003/arcrole/summation-item"
  xlink:from="concept_nr_employees_total"
  xlink:to="concept_nr_employees_age_41_and_up"
  weight="1.0"
  order="3" />
```

The *arcrole* is set to 'summation-item' which indicates that the 'from' concept is a summation calculated with items in the 'to' concept.

The attribute of interest here is *weight* which specifies the factor to multiply with when adding an item to the summation. In our case we have a simple addition, so all weights are 1.0.

6.3.3 What's the grand total?

When we render the instance document with our simple rendering tool, the following is shown:

Sample Report Viewer

Instance : sample_instance-2006-01-03.xml
Taxonomy : sample-2006-01-03.xsd

Age related demographics on employees

Entity : 12-34567

	2005-01-01	2005-12-31
Total number employees	35	41*
.. - 20	5	9
21 - 40	23	21
41 - ..	7	11

- SummationItemNetwork.validate.1: Consistency Check Failure for Total number employees on Gender related demographics on employees:
reported: 41
expected: 42.

Gender related demographics on employees

Entity : 12-34567

	2005-01-01	2005-12-31
Total number employees	35	41*
Male	23	27
Female	12	15

- SummationItemNetwork.validate.1: Consistency Check Failure for Total number employees on Gender related demographics on employees:
reported: 41
expected: 42.

The calculation on 'Gender related demographics' resulted in a validation error: the reported value was 41, but the calculated value was 42. Just like we would expect.

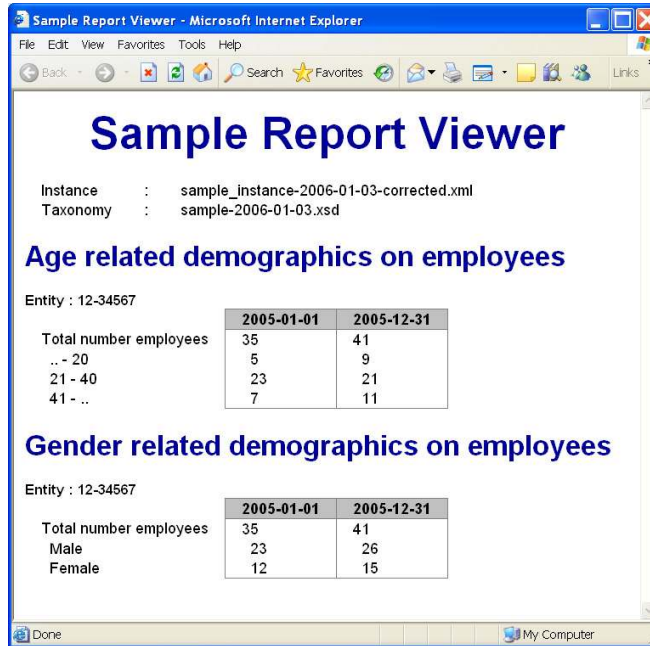
One limitation of calculations is shown here as well: our rendering tool couldn't decide which occurrence of the fact to show the error on. The 'Total number employees' fact for that concept is used in both sections, so the error is shown in both sections.

In this case, a more clever rendering tool might detect that the role is that of 'Gender related demographics' and only show the error in the section with the same role. But this is not really a solution, since we could just as easily have used new custom roles for the calculation linkbase. In that case the rendering tool would have no way to determine where to show the error.

At this moment, the only solution seems to be to use separate concepts for the total number of employees in age or gender related demographics. That would however defeat the purpose to try to report facts only once.

6.3.3.1 CORRECTION

If we correct the report, changing the number of male employees to 26, the rendered report finds that everything is fine again.



The screenshot shows a web browser window titled "Sample Report Viewer - Microsoft Internet Explorer". The page content includes:

- Instance : sample_instance-2006-01-03-corrected.xml
- Taxonomy : sample-2006-01-03.xsd
- Age related demographics on employees**
- Entity : 12-34567
- Table with columns for 2005-01-01 and 2005-12-31, and rows for Total number employees, .. - 20, 21 - 40, and 41 - ..
- Gender related demographics on employees**
- Entity : 12-34567
- Table with columns for 2005-01-01 and 2005-12-31, and rows for Total number employees, Male, and Female.

	2005-01-01	2005-12-31
Total number employees	35	41
.. - 20	5	9
21 - 40	23	21
41 - ..	7	11

	2005-01-01	2005-12-31
Total number employees	35	41
Male	23	26
Female	12	15

6.4 Day 4 – How To Make Life Easier

Up till now we've been using only item concepts. Every different nr_employees concept can be interpreted on its own.

There is however a drawback to this approach: the different types of gender will usually stay limited to two. For age we might however have a lot more groups instead of the three in our example.

What if we needed to divide the ages into 10-year intervals? We would have to include concepts for groups 11-20; 21-30; 31-40; 41-50; 51-60 & 61-70.

When dividing in 5-year intervals, 2-year intervals or even one group per number of years, this quickly becomes very tedious. The linkbases would become very large, simply because we would have to include links for every concept separately. We would like to have a more efficient way to do this.

Today we will look at a way to use tuples to solve this problem.



Note This is not really a good example.

Categorizations of concepts, such as by gender and by age group should be captured in so-called dimensional taxonomies as shown in the next sections.

6.4.1 Structural Changes

The problem we have arises because we have included the category (gender / age group) in the concept of number of employees. What if we would separate the two into different concepts?

This would lead to the following concept definitions:

```
<element
  id="sample_nr_employees_total"
  name="nr_employees_total"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />
<element
  id="sample_nr_employees"
  name="nr_employees"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />
<element
  id="sample_gender"
  name="gender"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />
<element
  id="sample_age_group"
  name="age_group"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />
```

This time around, most of our concepts can no longer be interpreted independently: what does a *gender* mean and what does a *nr_employees* mean?

Enter our tuples: the combination of *gender* and *nr_employees* and the combination of *age_group* with *nr_employees* does lead to a meaningful concept.

```
<element
  id="sample_gender_data"
  name="gender_data"
  substitutionGroup="xbrli:tuple"
  nillable="true">
  <complexType>
    <sequence>
      <element ref="sample:gender"/>
      <element ref="sample:nr_employees"/>
    </sequence>
  </complexType>
</element>
<element
  id="sample_age_group_data"
  name="age_group_data"
  substitutionGroup="xbrli:tuple"
  nillable="true">
  <complexType>
    <sequence>
      <element ref="sample:age_group"/>
      <element ref="sample:nr_employees"/>
    </sequence>
  </complexType>
</element>
```

A tuple is an element with *substitutionGroup*'tuple'. It has a complex type in which a sequence of other concepts is given.

We have to update the label linkbase and presentation linkbase to use the new concepts accordingly.

The presentation links would look something like this:

```
<presentationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/genderDemographics" >

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_nr_employees_total"
    xlink:label="concept_nr_employees_total" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_gender_data"
    xlink:label="concept_gender_data" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_gender"
    xlink:label="concept_gender" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_nr_employees"
    xlink:label="concept_nr_employees" />

  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_gender_data"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_gender_data"
    xlink:to="concept_gender"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_gender_data"
    xlink:to="concept_nr_employees"
    order="2" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>

<presentationLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/ageDemographics" >

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_nr_employees_total"
    xlink:label="concept_nr_employees_total" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_age_group_data"
    xlink:label="concept_age_group_data" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_age_group"
    xlink:label="concept_age_group" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-04.xsd#sample_nr_employees"
    xlink:label="concept_nr_employees" />

  <presentationArc
    xlink:type="arc"
    xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
    xlink:from="concept_nr_employees_total"
    xlink:to="concept_age_group_data"
    order="1" priority="0" use="optional"
    preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
  <presentationArc
```

```

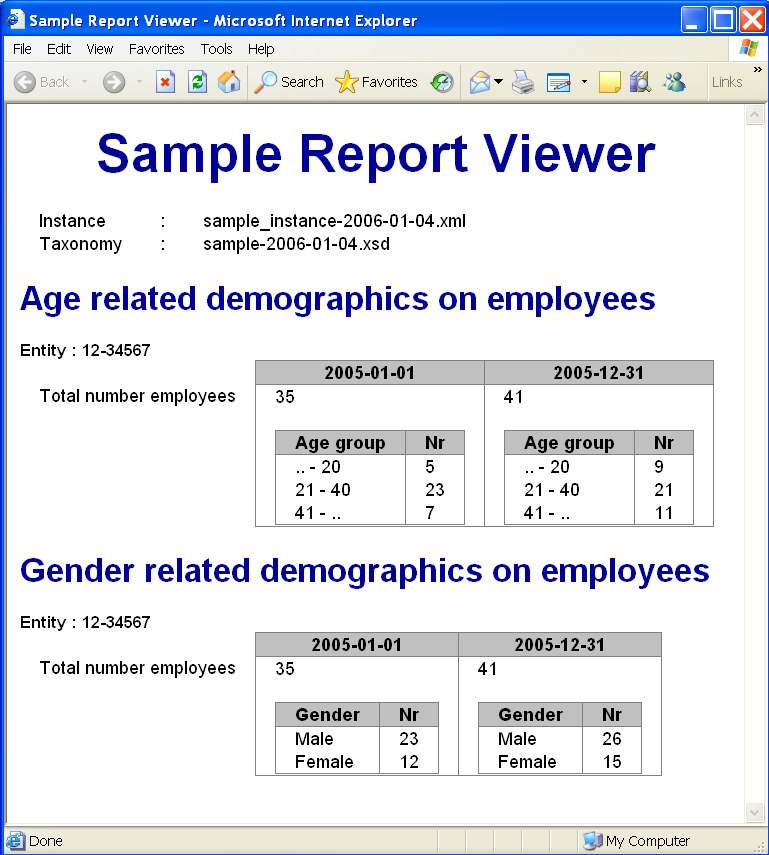
xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
xlink:from="concept_age_group_data"
xlink:to="concept_age_group"
order="1" priority="0" use="optional"
preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
<presentationArc
xlink:type="arc"
xlink:arcrole="http://www.xbrl.org/2003/arcrole/parent-child"
xlink:from="concept_age_group_data"
xlink:to="concept_nr_employees"
order="2" priority="0" use="optional"
preferredLabel="http://www.xbrl.org/2003/role/terseLabel" />
</presentationLink>

```

The tuples are placed as child below the total concept. Each tuple gets its child elements as children in the presentation hierarchy.

6.4.2 So What Does It Look Like?

Our simple renderer doesn't make assumptions on which item within the tuple could be seen as 'key', so it will simply create a little table of items, with the columns ordered according to the presentation linkbase.



Note that although our renderer does an OK job showing simple tuples, in general this can be a very difficult task. The items in tuples are not required to refer to the same context. And tuples can be infinitely nested, although that will probably never occur in practice.

This makes rendering tuples very difficult and will usually lead to custom rendering where knowledge of what is to be rendered can be used to decide how to layout tuples.

6.5 Day 5 – Opening up new Dimensions

In the previous sections we had reasonable success in translating our example form to XBRL, but it didn't quite fit. Today one of our designers had a brainwave: let's try out the XBRL Dimensions extension!

Looking at chapter five, it becomes immediately obvious that both gender and age group may be designed as dimensions. The taxonomy used up till now is 'promoted' to Primary Taxonomy *sample-2006-01-05.xsd*. We create a combined Domain Members and Template Taxonomy in *sample-2006-01-05-dimension.xsd*.

6.5.1 The Primary Concepts

As you might expect, we can simplify the sample taxonomy we have been using up till now. The primary concept that is needed is *nr_employees*. However, we will be using this concept within different dimensional contexts and we want to report the total number of employees in a context without dimensions. For this reason we also define the concepts *nr_employees_by_gender*, *nr_employees_by_age* and *nr_employees_total* in the *nr_employees* substitution group.

```
<element
  id="sample_nr_employees"
  name="nr_employees"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />

<element
  id="sample_nr_employees_by_gender"
  name="nr_employees_by_gender"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="sample:nr_employees"
  nillable="true" />

<element
  id="sample_nr_employees_by_age"
  name="nr_employees_by_age"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="sample:nr_employees"
  nillable="true" />

<element
  id="sample_nr_employees_total"
  name="nr_employees_total"
  xbrli:periodType="instant"
  type="xbrli:nonNegativeIntegerItemType"
  substitutionGroup="sample:nr_employees"
  nillable="true" />
```

The facts about the number of employees will simply be reported in several dimensions, using the different substitutions of *nr_employees* to keep the dimensions (and presentations) separated.

6.5.2 A Template Taxonomy

The Domain Members and Template Taxonomy are combined into one taxonomy schema. The Template Taxonomy imports a number of other schemas:

```
<import
  namespace="http://www.xbrl.org/2003/instance"
  schemaLocation="http://www.xbrl.org/2003/xbrl-instance-2003-12-31.xsd" />
<import
  namespace="http://xbrl.org/2005/xbrldt"
  schemaLocation="http://www.xbrl.org/2005/xbrldt-2005.xsd" />
<import
  namespace="http://xbrl.org/2006/xbrldi"
  schemaLocation="http://www.xbrl.org/2006/xbrldi-2006.xsd" />
```

```

<import
  namespace="http://www.sample.com/2006-01-05"
  schemaLocation="http://www.sample.com/2006-01-05/sample-2006-01-05.xsd" />

```

The xbrldi schema is not used in the Template Taxonomy itself, but it is used in the instance document, which in turn refers to the Template Taxonomy. To keep the dependency of instance documents on the dimensional module as limited as possible, the Template Taxonomy does the import. The Template Taxonomy must also refer to the Primary Taxonomy by importing it, to be able to reference primary items.

6.5.3 A Typical Dimension

As we know, there are two kinds of dimensions: typed and explicit. This section will look at how 'age group' could be set up as a typed dimension.

First of all, we need an element to define an XML Schema type for age_group. We'll use a simple type with an enumeration of possible values:

```

<element
  id="sample_age_group_type"
  name="age_group_type">
  <simpleType>
    <restriction base="string">
      <enumeration value=".. - 20" />
      <enumeration value="21 - 40" />
      <enumeration value="41 - .." />
    </restriction>
  </simpleType>
</element>

```

Having the type, we can declare the age group dimension which refers to the type in its *typedDomainRef* attribute.

```

<element
  id="sample_dim_age_group"
  name="dim_age_group"
  substitutionGroup="xbrldt:dimensionItem"
  type="xbrli:stringItemType"
  abstract="true"
  xbrli:periodType="instant"
  xbrldt:typedDomainRef="#sample_age_group_type" />

```

Note that the element is defined abstract, which is a requirement. The substitution group is *xbrldt:dimensionItem*.

6.5.4 Watch out: Explicit Language!

Just for the fun of it, this section will show the gender set up as an explicit dimension. This means we have to declare a set of items for the members as well as the dimension.

```

<element
  id="sample_dim_gender"
  name="dim_gender"
  substitutionGroup="xbrldt:dimensionItem"
  type="xbrli:stringItemType"
  abstract="true"
  xbrli:periodType="instant" />

<element
  id="sample_gender"
  name="gender"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />

<element
  id="sample_male"
  name="male"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />

```

```

<element
  id="sample_female"
  name="female"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:item"
  nillable="true" />

```

Note that the *dimensionItem* element is defined abstract, which is a requirement.

The elements meant for dimension members all are in the *xbrli:item* substitution group.

The *gender* element serves as the root of the dimension member hierarchy. The other elements are the members in the domain.

The members are assigned in the hierarchy with *domain-member* relationships, for which we create a definition linkbase. The linkbase will use a role to group the relationships and it is referenced from the taxonomy schema.

```

<link:roleType
  roleURI="http://www.sample.com/genderDimension" id="genderDimension">
  <link:definition>Gender dimension for demographics on employees
</link:definition>
  <link:usedOn>link:definitionLink</link:usedOn>
</link:roleType>
<link:linkbaseRef
  xlink:type="simple"
  xlink:href="sample-2006-01-05-dimension-definition.xml"
  xlink:role="http://www.xbrl.org/2003/role/definitionLinkbaseRef"
  xlink:arcrole="http://www.w3.org/1999/xlink/properties/linkbase" />

```

Within the linkbase we reference the role and locate the member items as usual.

```

<roleRef
  xlink:type="simple"
  xlink:href="sample-2006-01-05-dimension.xsd#genderDimension"
  roleURI="http://www.sample.com/genderDimension" />

<definitionLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/genderDimension" >
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-05-dimension.xsd#sample_dimension_gender"
    xlink:label="dimension_gender" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-05-dimension.xsd#sample_gender"
    xlink:label="domain_gender" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-05-dimension.xsd#sample_male"
    xlink:label="member_male" />
  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-05-dimension.xsd#sample_female"
    xlink:label="member_female" />
</definitionLink>

```

The dimension is linked to its domain by a *dimension-domain* relationship. The members of the domain are defined by *domain-member* relationships between the located items in the definition link:

```

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/int/dim/arcrole/dimension-domain"
  xlink:from="dimension_gender"
  xlink:to="domain_gender"
  order="1" priority="0" use="optional" />
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/int/dim/arcrole/domain-member"
  xlink:from="domain_gender"
  xlink:to="member_male"
  order="1" priority="0" use="optional" />

```

```

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://www.xbrl.org/int/dim/arcrole/domain-member"
  xlink:from="domain_gender"
  xlink:to="member_female"
  order="2" priority="0" use="optional" />

```

6.5.5 What's the Hype?

Now that we have a Primary Taxonomy and a Domain Member Taxonomy, we can combine these with hypercubes in a Template Taxonomy.

First we declare the hypercube concepts themselves as abstract elements in the *hypercubeItem* substitution group.

```

<element
  id="sample_hc_gender"
  name="hc_gender"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:hypercubeItem"
  nillable="true"
  abstract="true" />

<element
  id="sample_hc_age_group"
  name="hc_age_group"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrli:hypercubeItem"
  nillable="true"
  abstract="true" />

```

The hypercubes and primary item concept are located in the definition linkbase and relationships are created from primary concept to hypercube to dimension.

```

<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-05.xsd#sample_nr_employees_by_gender"
  xlink:label="primary_nr_employees_by_gender" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-05-dimension.xsd#sample_hc_gender"
  xlink:label="hypercube_gender" />

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/all"
  xbrldt:contextElement="scenario"
  xlink:from="primary_nr_employees_by_gender"
  xlink:to="hypercube_gender"
  order="1" priority="0" use="optional" />

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_gender"
  xlink:to="dimension_gender"
  order="1" priority="0" use="optional" />

```

For the age group dimension we make a similar definition link that locates the concepts and creates arc relationships between the primary concept, the hypercube and the dimension.

```

<definitionLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/ageGroupDimension" >

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-05.xsd#sample_nr_employees_by_age"
    xlink:label="primary_nr_employees_by_age" />

```

```

<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-05-dimension.xsd#sample_hc_age_group"
  xlink:label="hypercube_age_group" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-05-dimension.xsd#sample_dim_age_group"
  xlink:label="dimension_age_group" />

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/all"
  xbrl:contextElement="scenario"
  xlink:from="primary_nr_employees_by_age"
  xlink:to="hypercube_age_group"
  order="1" priority="0" use="optional" />

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_age_group"
  xlink:to="dimension_age_group"
  order="1" priority="0" use="optional" />
</definitionLink>

```

Note that the link uses another role defined in the dimension taxonomy: *ageGroupDimension*:

```

<link:roleType
  roleURI="http://www.sample.com/genderDimension" id="ageGroupDimension">
  <link:definition>Age group dimension for demographics on employees
  </link:definition>
  <link:usedOn>link:definitionLink</link:usedOn>
</link:roleType>

```

6.5.6 Presenting in dimensions

The presentation and label linkbases are similar to the ones we have been using before, we won't be rehashing them here.

6.5.7 Dimensional instance

Now that we have our dimensional taxonomy set up, we can make an instance document using the dimensions. First we define contexts for the total number of employees both at the start and the end of the reporting period.

```

<context id="c_start">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
</context>
<context id="c_end">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
</context>

```

Next we define contexts for the values of the age group dimension.

```

<context id="c_start_age_up_to_20">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>

```



```

<period>
  <instant>2005-01-01</instant>
</period>
<scenario>
  <xbrldi:typedMember dimension="sd:dim_age_group">
    <sd:age_group_type>.. - 20</sd:age_group_type>
  </xbrldi:typedMember>
</scenario>
</context>
<context id="c_end_age_up_to_20">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>.. - 20</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>

<context id="c_start_age_21_to_40">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>21 - 40</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>
<context id="c_end_age_21_to_40">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>21 - 40</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>

<context id="c_start_age_41_and_up">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>41 - ..</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>
<context id="c_end_age_41_and_up">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>41 - ..</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>

```

The dimension is referred to in the *dimension* attribute of the *xbrldi:typedMember* element. The value for the dimension is specified as a *sd:age_group_type* element, which is the declared type of the dimension, with the correct value as content.

Next we define contexts for the values of the gender dimension.

```
<context id="c_start_gm">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:male</xbrldi:explicitMember>
    </scenario>
  </context>
<context id="c_end_gm">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:male</xbrldi:explicitMember>
    </scenario>
  </context>

<context id="c_start_gf">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:female</xbrldi:explicitMember>
    </scenario>
  </context>
<context id="c_end_gf">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:female</xbrldi:explicitMember>
    </scenario>
  </context>
```

The dimension is referred to in the *dimension* attribute of the *xbrldi:explicitMember* element. The value for the dimension is specified as its content, which corresponds to the element name of the dimension domain member.

Note that the instance document references the Template Taxonomy, instead of the Primary:

```
<link:schemaRef
  xlink:type="simple"
  xlink:href="http://www.sample.com/2006-01-05/sample-2006-01-05-dimension.xsd" />
```

Finally we can report the facts, each within it's own context.

```
<!-- totals, in dimensionless context -->
<s:nr_employees_total
  contextRef="c_start"
  unitRef="u_person"
  decimals="0">35</s:nr_employees_total>
<s:nr_employees_total
  contextRef="c_end"
  unitRef="u_person"
  decimals="0">41</s:nr_employees_total>

<!-- Gender Demographics -->
<s:nr_employees_by_gender
  contextRef="c_start_gm"
  unitRef="u_person"
  decimals="0">23</s:nr_employees_by_gender>
<s:nr_employees_by_gender
  contextRef="c_end_gm"
  unitRef="u_person"
  decimals="0">27</s:nr_employees_by_gender>

<s:nr_employees_by_gender
  contextRef="c_start_gf"
  unitRef="u_person"
  decimals="0">12</s:nr_employees_by_gender>
<s:nr_employees_by_gender
  contextRef="c_end_gf"
  unitRef="u_person"
  decimals="0">15</s:nr_employees_by_gender>

<!-- Age Group Demographics -->
<s:nr_employees_by_age
  contextRef="c_start_age_up_to_20"
  unitRef="u_person"
  decimals="0">5</s:nr_employees_by_age>
<s:nr_employees_by_age
  contextRef="c_end_age_up_to_20"
  unitRef="u_person"
  decimals="0">9</s:nr_employees_by_age>

<s:nr_employees_by_age
  contextRef="c_start_age_21_to_40"
  unitRef="u_person"
  decimals="0">23</s:nr_employees_by_age>
<s:nr_employees_by_age
  contextRef="c_end_age_21_to_40"
  unitRef="u_person"
  decimals="0">21</s:nr_employees_by_age>

<s:nr_employees_by_age
  contextRef="c_start_age_41_and_up"
  unitRef="u_person"
  decimals="0">7</s:nr_employees_by_age>
<s:nr_employees_by_age
  contextRef="c_end_age_41_and_up"
  unitRef="u_person"
  decimals="0">11</s:nr_employees_by_age>
```

6.5.8 Our first peek in another dimension

The simple renderer we use has no support for dimensions, so the report looks like this:

The screenshot shows a web browser window titled 'Sample Report Viewer - Microsoft Internet Explorer'. The page content includes:

Instance : sample_instance-2006-01-05.xml
 Taxonomy : sample-2006-01-05.xsd

Age related demographics on employees

Entity : 12-34567

	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-12-31	2005-12-31	2005-12-31	2005-12-31	2005-12-31	2005-12-31
Total number employees						35						41
Number employees		5	7	23				9	11	21		

Gender related demographics on employees

Entity : 12-34567

	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-01-01	2005-12-31	2005-12-31	2005-12-31	2005-12-31	2005-12-31	2005-12-31
Total number employees						35						41
Number employees	23	12					27	15				

After some manual changes, removing unnecessary columns, resorting the remaining columns and adding another header row we can have a look at a somewhat better presentation as you might expect from a dimensions aware renderer:

The screenshot shows a web browser window titled 'Sample Report Viewer - Microsoft Internet Explorer'. The page content includes:

Instance : sample_instance-2006-01-05.xml
 Taxonomy : sample-2006-01-05.xsd

Age related demographics on employees

Entity : 12-34567

	2005-01-01	By age			2005-12-31	By age		
	Total	.. - 20	21 - 40	41 - ..	Total	.. - 20	21 - 40	41 - ..
Total number employees	35				41			
Number employees		5	23	7		9	21	11

Gender related demographics on employees

Entity : 12-34567

	2005-01-01	By gender		2005-12-31	By gender	
	Total	Male	Female	Total	Male	Female
Total number employees	35			41		
Number employees		23	12		27	15

6.6 Day 6 – Entering Multiple Dimensions

Hypercubes with just one dimension aren't really worth the trouble, are they? So this last day we'll see what the dimensional taxonomy and instance document look like when we combine the two dimensions into one hypercube.

Doing this involves changes to all parts of the taxonomy, mostly dropping things we no longer need or replacing separate elements with one combined element:

- **Primary Concepts**
We will not distinguish between the two dimensions so we can drop the *nr_employees_by_age* and *nr_employees_by_gender* substitutions.
- **Dimension Taxonomy**
Since we can use the *gender* dimension member element to signify 'all genders', we add an 'all' token to the *age_group_type* to even things up.

```
<element
  id="sample_age_group_type"
  name="age_group_type">
  <simpleType>
    <restriction base="string">
      <enumeration value="all" />
      <enumeration value=".. - 20" />
      <enumeration value="21 - 40" />
      <enumeration value="41 - .." />
    </restriction>
  </simpleType>
</element>
```

- **Template Taxonomy**
We can replace the *hc_age_group* and *hc_gender* hypercubes with one *hc_gender_x_age_group* hypercube.

```
<element
  id="sample_hc_gender_x_age_group"
  name="hc_gender_x_age_group"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrldt:hypercubeItem"
  nillable="true"
  abstract="true" />
```

6.6.1 Definition Linkbase

It becomes interesting when we start looking at the definition linkbase.

First of all, the relationships from primary item to hypercube and from hypercube to dimension are removed from the *gender definitionLink*, leaving only the relationships between dimension and its members.

The age group *definitionLink* may be removed completely, replacing it with a *definitionLink* for the new hypercube.

```
<definitionLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/demographics" >

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-06.xsd#sample_nr_employees"
    xlink:label="primary_nr_employees" />

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-06-dimension.xsd#sample_hc_gender_x_age_group"
    xlink:label="hypercube_gender_x_age_group" />

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-06-dimension.xsd#sample_dim_gender"
    xlink:label="dimension_gender" />

  <loc
    xlink:type="locator"
    xlink:href="sample-2006-01-06-dimension.xsd#sample_dim_age_group"
    xlink:label="dimension_age_group" />
```

```

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/all"
  xbrldt:contextElement="scenario"
  xlink:from="primary_nr_employees"
  xlink:to="hypercube_gender_x_age_group"
  order="1" priority="0" use="optional" />

<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_gender_x_age_group"
  xlink:to="dimension_gender"
  xbrldt:targetRole="http://www.sample.com/genderDimension"
  order="1" priority="0" use="optional" />
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_gender_x_age_group"
  xlink:to="dimension_age_group"
  order="2" priority="0" use="optional" />
</definitionLink>

```

Note that the *hypercube-dimension* arcs are ordered.

Also note the *targetRole* attribute on the arc to the gender dimension refers to the role in which the gender dimension is declared. This allows the processor to go from one base set to the other when composing the DRS.

6.6.2 Instance document

The contexts of the instance document need to be changed to provide values for both dimensions. The following sample shows some possible contexts (it takes too much trees to print them all).

```

<context id="c_start_gm">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:male</xbrldi:explicitMember>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>all</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>

<context id="c_end_gf_age_up_to_20">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-12-31</instant>
  </period>
  <scenario>
    <xbrldi:explicitMember dimension="sd:dim_gender">
      sd:female</xbrldi:explicitMember>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>.. - 20</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>

```

The first context can be used to report the number of male employees at the start of the reporting period, regardless of age. Reporting the number of female employees aged up to 20 at the end of the reporting period can be done with the second context.

Note that the order for dimensions must be the same as specified in the definition linkbase.

Rendering this instance could be done in a different layout than that supported by our simple renderer in a 'matrix' like this (disregarding the totals):

Number of employees by age group and gender			
2005-01-01			
	... - 20	21 - 40	41 - ...
Male	4	15	4
Female	1	8	3
2005-12-31			
	... - 20	21 - 40	41 - ...
Male	6	14	7
Female	3	8	4

6.6.3 When in doubt, use a Default

Let's see if we can make life a little bit easier for reporters that want to report on age-group regardless of gender. Remember that the reporter can use the *domain_gender* domain member to signify 'all genders'. We can help the reporter by specifying the *domain_gender* dimension member as default:

```
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/dimension-default"
  xlink:from="dimension_gender"
  xlink:to="domain_gender"
  order="1" priority="0" use="optional" />
```

With this default in place, we can create a context like this:

```
<context id="c_start_age_up_to_20">
  <entity>
    <identifier scheme="http://www.statistics.org">12-34567</identifier>
  </entity>
  <period>
    <instant>2005-01-01</instant>
  </period>
  <scenario>
    <xbrldi:typedMember dimension="sd:dim_age_group">
      <sd:age_group_type>.. - 20</sd:age_group_type>
    </xbrldi:typedMember>
  </scenario>
</context>
```

A dimensions conformant processor will derive the default *domain_gender* value automatically.

6.6.4 Never Ask A Lady About Her Age

One final twist to our dimensional taxonomy is to exclude certain combinations of dimensions. For the sake of argument, suppose we are real gentlemen who don't ask a lady about her age. We can define a role and hypercube like this:

```
<link:roleType
  roleURI="http://www.sample.com/gentleman" id="gentleman">
  <link:definition>A gentlemen never asks a lady about her age
  </link:definition>
  <link:usedOn>link:definitionLink</link:usedOn>
```

```
<element
  id="sample_hc_female_x_age_group"
  name="hc_female_x_age_group"
  xbrli:periodType="instant"
  type="xbrli:stringItemType"
  substitutionGroup="xbrldt:hypercubeItem"
  nillable="true"
```

```
abstract="true" />
```

The hypercube is given it's own dimensions that include the female gender and all age groups, all within the *gentleman* role to keep it separate from the hypercube we already defined.

```
<definitionLink
  xlink:type="extended"
  xlink:role="http://www.sample.com/gentleman" >
</definitionLink>
```

Again we locate the elements we're going to need.

```
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-06.xsd#sample_nr_employees"
  xlink:label="primary_nr_employees" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-06-dimension.xsd#sample_hc_female_x_age_group"
  xlink:label="hypercube_female_x_age_group" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-06-dimension.xsd#sample_dim_gender"
  xlink:label="dimension_gender" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-05-dimension.xsd#sample_female"
  xlink:label="member_female" />
<loc
  xlink:type="locator"
  xlink:href="sample-2006-01-06-dimension.xsd#sample_dim_age_group"
  xlink:label="dimension_age_group" />
```

The new *female_x_age_group* hypercube is given the same two dimensions as the *gender_x_age_group* hypercube.

```
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_female_x_age_group"
  xlink:to="dimension_gender"
  order="1" priority="0" use="optional" />
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/hypercube-dimension"
  xlink:from="hypercube_female_x_age_group"
  xlink:to="dimension_age_group"
  xbrldt:targetRole="http://www.sample.com/ageGroupDimension"
  order="2" priority="0" use="optional" />
```

But the domain of the gender dimension consists of *female* only.

```
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/dimension-domain"
  xlink:from="dimension_gender"
  xlink:to="member_female"
  order="1" priority="0" use="optional" />
```

This time we make a *notAll* relationship between our familiar *nr_employees* primary item and the new hypercube to signify that the combinations in that hypercube are not allowed. This effectively prevents reports on any female - age group combination.

```
<definitionArc
  xlink:type="arc"
  xlink:arcrole="http://xbrl.org/int/dim/arcrole/notAll"
  xbrldt:contextElement="scenario"
  xlink:from="primary_nr_employees"
  xlink:to="hypercube_female_x_age_group"
  order="1" priority="0" use="optional" />

<!-- link hypercube to dimensions -->
```


6.7 Conclusion

This concludes our ventures into the XBRL language. As shown in this chapter, a lot is possible with XBRL, but you need to carefully consider how to set up a taxonomy and its linkbases to stay within the limits imposed by XBRL.